Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

A study on the effects of normalized TSP features for automated algorithm selection ☆

Jonathan Heins [a,∗], Jakob Bossek [b], Janina Pohl [c], Moritz Seiler [c], Heike Trautmann [c], Pascal Kerschke [a]

[a] *Big Data Analytics in Transportation TU Dresden, Dresden, Germany*
[b] *AI Methodology, RWTH Aachen University, Aachen, Germany*
[c] *Statistics and Optimization, University of Muenster, Muenster, Germany*

A R T I C L E   I N F O

Article history:
Received 31 March 2022
Received in revised form 6 October 2022
Accepted 11 October 2022
Available online xxxx

Keywords:
Feature normalization
Algorithm selection
Traveling salesperson problem

A B S T R A C T

Classic automated algorithm selection (AS) for (combinatorial) optimization problems heavily relies on so-called instance features, i.e., numerical characteristics of the problem at hand ideally extracted with computationally low-demanding routines. For the traveling salesperson problem (TSP) a plethora of features have been suggested. Most of these features are, if at all, only normalized imprecisely raising the issue of feature values being strongly affected by the instance size. Such artifacts may have detrimental effects on algorithm selection models. We propose a normalization for two feature groups which stood out in multiple AS studies on the TSP: (a) features based on a minimum spanning tree (MST) and (b) nearest neighbor relationships of the input instance. To this end we theoretically derive minimum and maximum values for properties of MSTs and k-nearest neighbor graphs (NNG) of Euclidean graphs. We analyze the differences in feature space between normalized versions of these features and their unnormalized counterparts. Our empirical investigations on various TSP benchmark sets point out that the feature scaling succeeds in eliminating the effect of the instance size. A proof-of-concept AS-study shows promising results: models trained with normalized features tend to outperform those trained with the respective vanilla features.

1. Introduction

The Euclidean traveling salesperson problem (TSP) is a prominent NP-hard combinatorial optimization problem of huge practical relevance, e.g., for the fabrication of printed circuit boards as well as for transportation and logistics applications. The problem can be described as follows: we are given a complete undirected graph $G = (V, E)$ with $n = |V|$ nodes, node set $V = \{1, \dots, n\}$ and pairwise distances $d(i, j), 1 \leq i, j \leq n$ between the nodes which are determined by the Euclidean metric. The goal is to compute a tour of minimal length that visits each city exactly once and eventually returns to the start of the tour. Formally, we seek a permutation $\pi : V \to V$ that minimizes the cost function

☆ Conference version published at FOGA2021.
∗ Corresponding author.
E-mail address: jonathan.heins@tu-dresden.de (J. Heins).

https://doi.org/10.1016/j.tcs.2022.10.019
0304-3975/© 2022 Elsevier B.V. All rights reserved.

$$c(\pi) = d(\pi(n), \pi(1)) + \sum_{i=1}^{n-1} d(\pi(i), \pi(i+1)).$$

The TSP has been an intriguing problem in research for decades. It has ever since served as a test-bed for algorithmic ideas ultimately culminating in the development of the highly sophisticated exact TSP solver concorde [1] and several very well-performing heuristics such as the local-search algorithm LKH [12] and the genetic algorithm EAX [23,24].

Both algorithms perform well on different TSP instances potentially with a large performance gap to each other [8] which motivates per-instance automated algorithm selection (AS) for the TSP. In the past decade, during the rise of machine learning and more precisely AS-related research, the TSP continued to attract researchers [4,5,8,9,14,17,19,22,25,27,33,34]. In the context of AS, initiated by Rice in the 1970s [30], the core idea is to predict – by means of machine learning models termed *selectors* in this field – the most likely best-performing solver for a given problem instance $I \in \mathcal{I}$ from a set $\mathcal{A}$ of at least two candidate solvers (see [16,18] for recent surveys on AS). In classic AS approaches the instance space $\mathcal{I}$ in the selector-mapping $S : \mathcal{I} \to \mathcal{A}$ is replaced by a p-dimensional feature-space $\mathcal{F} \subset \mathbb{R}^p$, and each instance is mapped to the feature space by a computationally low-demanding function $f : \mathcal{I} \to \mathcal{F}$. The selector can be replaced by $S : \mathcal{F} \to \mathcal{A}$.

In order to numerically characterize TSP problems, hundreds of features were developed by several groups [14,22,27]. Exemplary features that stand out in several studies as crucial performance-discriminating features, are based on properties of a minimum spanning tree (MST) of the graph at hand (e.g., summary statistics of the depth of nodes in the MST) or statistics of nearest neighbor relationships including weakly/strongly connected components (CCs) of a so-called k-nearest-neighbor graph transformation (k-NNGs) of the source graph, e.g., the number of CCs or the size of the largest CC. However, until now, those features were used in an unnormalized way which may be detrimental and is certainly against intuition. As an example say we have two graphs G_1 with 30 nodes and G_2 with 3 000 nodes respectively. The number of weakly connected components in the 2-NNG may be up to 10 for G_1 and 1 000 for G_2 as we shall see later, although the pattern of weakly connected components in the 2-NNG is the same for both graphs, i.e., a collection of 3-cliques.[1] Such artifacts may induce a negative impact on algorithm selection due to features being dependent on the instance size. This motivated the conference version of this work [11] and its extension presented in this paper. The aim is to introduce normalized versions of established TSP features

$$f_{\text{norm}} : \mathcal{I} \to \mathcal{F}_{\text{norm}} \subseteq [0,1]^p.$$

The respective normalization is based on the theoretical derivation of minimum and maximum values of MST- and nearest neighbor relationship features which proved to have considerable discriminating impact on AS models in previous works [17, 27]. This is a substantial and mathematically sound extension of preliminary normalization efforts [27], in which the feature values have been divided by the instance size.

We will show that our normalized features not only improve the understanding of feature and algorithm performance relations but also allow for constructing high-performing AS models on the Euclidean TSP across instance sizes. Section 2 provides the theoretical concept of our normalization approach together with the respective normalized feature variants. Compared to the conference version, the considered feature groups are extended by distance based features in this paper. Following is an overview of the considered instance sets and inexact TSP solvers in Section 3. An exploratory illustration of the feature normalization effects on widely used and commonly accepted TSP instance sets [17] is presented in Section 4. A proof-of-concept AS study in Section 5 illustrates the promising potential of the introduced feature sets as input to automated algorithm selection models. Eventually, Section 6 summarizes the paper and discusses future work.

2. Normalizing features

In this section we derive theoretical results to normalize specific features. In abstract language, given a TSP feature value $f \in \mathbb{R}$ for a Euclidean graph G, we seek for lower and upper bounds $f_{\min}$ and $f_{\max}$ for the respective feature among all Euclidean graphs in order to calculate the *normalized* feature

$$f_{\text{norm}} := \frac{f - f_{\min}}{f_{\max} - f_{\min}} \in [0,1].$$

Deriving those upper and lower bounds of the features and normalizing with them is desired for two main reasons in addition to general interest. First, algorithmic performance should be independent of the scaling of distances within an instance, and therefore, selectors and the used features dependent on the distances should be independent as well. Contrarily, the algorithmic performance may differ depending on n, and therefore, the features should encode this information. However, this information can be represented by one feature: the instance size. Including the information in multiple features may introduce noise, complicating the task of a selector from a machine learning perspective. Nevertheless, in some cases the upper and lower bounds may scale differently compared to the mean feature value over the distribution of all

[1] A k-clique in a graph $G = (V, E)$ is a subset of nodes $C \subseteq V$, $|C| = k$ such that each two nodes in C are linked by an edge.

instances. In those cases, even a normalized feature value, encoding a certain property of an instance, may shrink or grow with n. Subtracting the mean feature value over the distribution of all instances for given n and dividing by the standard deviation would alleviate this problem. However, even if a uniform distribution of nodes in an instance could be assumed representative, deriving the expected feature values is far from trivial. Coarse normalization methods, e.g., dividing by the mean as done in [37,38], may be one approach to the problem. In the case of different scaling behaviors we deem the derivation of upper and lower bounds important for a second reason. In recent work instances were evolved with a space filling approach based on the upper and lower bounds [7]. This reveals new insights into what characteristics of an instance cause different difficulty levels for the solvers. Those insights, in turn, help understand the solver behavior and ultimately can result in better selectors.

For this study, we do not consider all 200+ TSP features from the literature for two simple reasons: (1) space limitations, and (2) we do not have formal proofs for all feature normalizations (yet). Nevertheless, we do emphasize that a normalization of all features is desirable in future. Here, we focus on two feature groups that have proven to be very promising in discriminating solver performance in past TSP algorithm selection studies [17,27], namely features based on a minimum spanning tree of the input graph (MST features) and the nearest neighbor relationship features (NN features). Implementations of the (normalized) features are available within the package `salesperson` [3] for the statistical programming language R [28].[2]

2.1. Preliminaries

A graph $G = (V, E)$ consists of a set of nodes $V = \{v_1, \dots, v_n\}$ and a set of edges E. For brevity we denote by $n := |V|$ and $m := |E|$ the respective sizes. In a directed graph each edge (u, v) between nodes $u, v \in V$ has an orientation. In contrast, in undirected graphs there is no orientation which is highlighted by the set-style notation $\{u, v\}$. Each node $v \in V$ is associated with Euclidean coordinates (x_v, y_v) in the Euclidean plane. A function $d : V \times V \to \mathbb{R}_{\geq 0}$ describes distances/travel costs between pairs of nodes. d is a pseudometric and referred to as distance function in the following. The distance $d(u, v)$ between two nodes $u, v \in V$ corresponds to the Euclidean distance of the node coordinates of u and v in the plane. A *directed path* is a sequence of nodes $v_1, \dots, v_l$ such that $(v_i, v_{i+1}) \in E$ for $1 \leq i < l$. In an undirected path, the corresponding undirected edges need to exist. For $U \subset V$ we define by $G[U] := (U, \{e \in E \mid |e \cap U| = 2\})$ the *induced subgraph* of G given U; $G[U]$ contains all nodes from U and all edges which link each two nodes in U. A graph is *(strongly) connected* if every node is reachable via a (directed) path from every other node. A subset of nodes $C \subset V$ is called a *strongly connected component* if the induced subgraph $G[C]$ is strongly connected and there is no superset of C that is strongly connected. Likewise, $C \subset V$ forms a *weakly connected component* if the induced subgraph is maximally weakly connected where the latter means that each node is reachable from any other node if edge orientations are neglected. A *spanning tree* of G is an acyclic subgraph of G covering all nodes. A *minimum spanning tree* (MST) is a spanning tree with minimum sum of edge weights across all possible spanning trees.

2.2. k-nearest-neighbor graph based features

An important group of TSP features relies on characteristics of local nearest-neighbor relations, i.e., nodes and their k nearest neighbor nodes.[3] This feature set was introduced by Pihera & Musliu [27] and is based on a k-NNG transformation of the input graph (k-NNG features). Below, we give a formal definition following the notation used in [27].

Definition 2.1 *(k-nearest-neighbor graph).* Let $G = (V, E)$ be a graph with distance function $d : V \times V \to \mathbb{R}_{\geq 0}$. For $1 \leq i \leq n - 1$ let $N(v, i)$ be the i-th nearest neighbor of node $v \in V$. Further denote by $E_k^d = \{(v, N(v, i)) \mid v \in V, 1 \leq i \leq k\}$ and $E_k^u = \{\{v, N(v, i)\} \mid v \in V, 1 \leq i \leq k\}$ the set of directed/undirected edges between nodes and their $1 \leq k \leq n - 1$ nearest neighbor nodes. Then we call $G_k^d = (V, E_k^d)$ and $G_k^u = (V, E_k^u)$ the directed/undirected *k-nearest-neighbor graph* (k-NNG).

Fig. 1 gives an impression of k-NNGs for different values of k on instance `a280` from the TSPLIB [29]. Given the directed and undirected k-NNG for a fixed k, this feature group consists of the number of weakly/strongly connected components and summary statistics of the number of nodes across weakly/strongly connected components, i.e., the mean, median, minimum, maximum and the span. It is obvious that these take values that highly depend on the graph size n. It should be noted that Pihera & Musliu also introduce normalized versions of these features, where normalization is achieved by dividing with instance size n. Since n is not the maximum value achievable for most of the features (see our derivations below) we do not consider this step as a sufficiently precise normalization. Therefore, we next derive results on the minimum and maximum possible values in order to normalize all NNG-features mathematically soundly.

[2] GitHub repository: https://github.com/jakobbossek/salesperson/.

[3] Note that internally the Approximate Near Neighbor (ANN) C++ library is used to construct a k-d tree in order to search for the nearest neighbors of each node. In the case of equidistant neighbors the later found neighbor is used. Thus, in this case the k-NNG depends on the construction of the k-d tree which may be different if, e.g., the instance is rotated. However, this does not influence the theoretical upper and lower bounds.

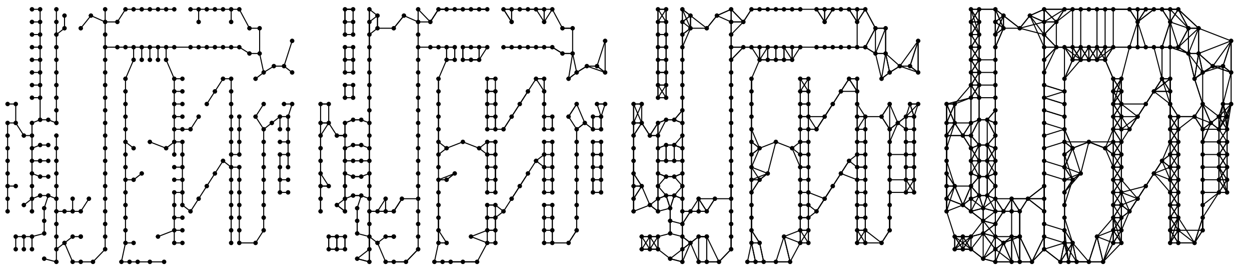

Fig. 1. From left to right: an MST, 1-NNG, 3-NNG and 5-NNG of TSPLIB instance `a280`.

Theorem 2.1. *For a complete undirected graph $G = (V, E)$ with $d : V \times V \to \mathbb{R}_{\geq 0}$ and $k \geq 2$ the minimal number of weakly/strongly connected components in G_k^d and G_k^u equals one.*

Proof. Consider a graph where nodes $v_1, \ldots, v_n$ are placed circular with equidistant distances between neighbor nodes. Then for $k = 2$, in G_k^d, each node is linked to its two neighbors on the cycle via directed edges. In consequence, in G_k^d each node can be reached from any other node, e.g., by following edges in clock-wise direction. Thus, the number of strongly connected components equals one and so does the number of weakly connected components. Adding further edges in case $k > 2$ does not change the connectivity and therefore the results transfer to the general case. □

Theorem 2.2. *For a complete undirected graph $G = (V, E)$ with $d : V \times V \to \mathbb{R}_{\geq 0}$ and $k \geq 2$ the maximum number of weakly connected components in G_k^u is $\lfloor n/(k + 1)\rfloor$.*

Proof. Each weakly connected component contains at least $k+1$ nodes since every node is linked to its k nearest neighbors and reachability is symmetric in undirected graphs. Thus, the number of weakly components is at most $\lfloor n/(k + 1)\rfloor$. □

An example for a Euclidean graph with a maximal number of weakly connected components is a graph with $\left\lfloor \frac{n}{k+1} \right\rfloor$ non-overlapping clusters where the inter-cluster distance is higher than the distance between any pair of nodes within a cluster.

The following lemma will prove useful to derive the maximum number of strongly connected components in G_k^d.

Lemma 2.1. *For any $0 < q < 1/2$, $L \geq 0$ and $k \geq 1$ it holds that $q^L > q^{L+1} + q^{L+2} + \ldots + q^{L+k}$.*

Proof. Let $L \geq 0$ and $k \geq 1$ and $0 < q < 1$. Then we have

$$\begin{aligned}\sum_{j=1}^{k} q^{L+j} &= q^L \left(\sum_{j=1}^{k} q^j\right) = q^L \left(\sum_{j=0}^{k} q^j - 1\right) \\ &= q^L \left(\frac{1-q^{k+1}}{1-q} - \frac{1-q}{1-q}\right) = q^L \left(\frac{q-q^{k+1}}{1-q}\right) \\ &= \frac{q^{L+1}\overbrace{(1-q^k)}^{<1}}{1-q} \leq \frac{q^{L+1}}{1-q}\end{aligned}$$

where we used the geometric sum $\sum_{j=0}^{n} q^j = \frac{1-q^{n+1}}{1-q}$ for $q \neq 1$ in the third equation. Eventually $q^{L+1}/(1-q) < q^L \Leftrightarrow q < 1-q \Leftrightarrow q < 1/2$. □

Theorem 2.3. *For a complete undirected graph $G = (V, E)$ with $d : V \times V \to \mathbb{R}_{\geq 0}$ and $k \geq 1$ the maximum number of strongly connected components in G_k^d equals $n - k$.*

Proof. We first construct a complete Euclidean graph $G = (V, E)$ with exactly $n - k$ strongly connected components in G_k^d and later argue, that more than $n - k$ components are impossible. Consider an n-vertex path where n nodes $v_1, \ldots, v_n$ are aligned from left to right. Let $C = 1/3$ be a constant. Place the nodes co-linear such that $d(v_i, v_{i+1}) = C^i$ for $1 \leq i < n$. In consequence, $d(v_1, v_2) > d(v_2, v_3) > \ldots > d(v_{n-1}, v_n)$. We show that for $1 \leq i \leq n - k$ the k nearest neighbors of v_i are $v_{i+1}, \ldots, v_{i+k}$. Since the distances $d(v_i, v_{i+1})$ are monotonically decreasing this holds if and only if $d(v_i, v_{i+k}) \leq d(v_{i-1}, v_i)$. We have

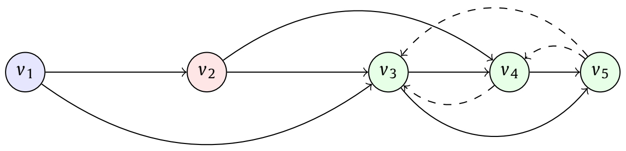

Fig. 2. Illustration of the 2-NNG graph of a 5-vertex path with strictly decreasing consecutive distances. Solid lines are forward edges, dashed lines are backward edges. Colors indicate membership to strongly connected components; in total there are $5 - 2 = 3$ here.

$$d(v_i, v_{i+k}) = \sum_{j=0}^{k-1} d(v_{i+j}, v_{i+j+1}) = \sum_{j=0}^{k-1} c^{i+j} < c^{i-1} = d(v_{i-1}, v_i)$$

where the first equality is due to the graph being Euclidean and the inequality follows from Lemma 2.1.

Hence, building the directed k-NNG up to node v_{n-k-1} does not introduce any cycles. However, for the latter nodes $v_{n-k+1}, \ldots, v_n$ there need to be right-to-left (backward) edges to fill up the set of k nearest neighbors which introduces directed cycles into the k-NNG up to node v_{n-k}. Hence, there is one component of size $n-(n-k-1) = k+1$ which results in a total number of $n-k$ strongly connected components. See Fig. 2 for a visualization with $k = 2$.

Eventually, we argue that there cannot exist a graph G where G_k^d has strictly more than $n-k$ strongly connected components. To this end it is sufficient to show that there has to be one component with at least $k+1$ nodes. Consider a k-NNG $G_k^d = (V_G, E_G)$ with strongly connected components containing at most $l < k+1$ nodes. For the nodes v_i in a component C of G_k^d with l or less nodes it holds that there are at least $k+1-l$ edges $(v_i, v_j) \in E_G$ with $v_j \notin C$. If each component in G_k^d is reduced to one node, the resulting graph is still directed and must also be acyclic, otherwise not all components have been reduced. However, each node of the new graph must have at least one outgoing edge, since there are fewer than $k+1$ nodes in each component. This leads to a contradiction, since an acyclic graph must have at least one sink. □

We are now ready to formulate lower and upper bounds for summary statistics of the number of nodes in weakly and strongly CCs of the k-NNG of Euclidean graphs.

Theorem 2.4. *For the number of nodes in the weakly connected components of the k-NNG G_k^u of any Euclidean graph with $d: V \times V \to \mathbb{R}_{\geq 0}$ and $\frac{n}{3} \geq k+1$, the following holds:*

1. *The minimum and maximum values of the mean are $\frac{n}{\lfloor n/(k+1) \rfloor}$ and n, respectively.*
2. *The minimum and maximum values of the median are $k+1$ and n, respectively.*
3. *The minimum and maximum values of the maximum are*
$$k+1+\left\lceil \frac{(n \mod (k+1))}{\lfloor n/(k+1) \rfloor} \right\rceil$$
and n, respectively.
4. *The minimum and maximum values of the minimum are $k+1$ and n, respectively.*
5. *The minimum and maximum values of the span[4] are 0 and $n-2(k+1)$, respectively.*

Proof. The upper bounds are trivial: all statistics except for the span take their maximal possible value n if there is just one single weakly CC containing all nodes; this is possible as shown in Theorem 2.1. The span takes its minimum, zero, in this case. Since the minimal possible number of nodes in a weakly connected component is $k+1$ the minimal possible number of nodes in a weakly connected component in G_k^u is $k+1$. Thus, $k+1$ is as well the minimal possible value for the median if there are cases with more than half of the components having $k+1$ nodes. This is satisfied if two components have $k+1$ nodes and a third contains all remaining nodes. Note that for $k+1 > \frac{n}{3}$ this is not possible since then only two strongly components can exist as a weakly CC has at least $k+1$ nodes. The mean number of nodes in a weakly CC is smallest if the nodes are equally distributed between the highest possible number of weakly connected components as described in Theorem 2.2 and therefore is $\frac{n}{\lfloor n/(k+1) \rfloor}$. Like the median the maximal number of nodes in a component cannot be smaller than $k+1$ as well and if n is not divisible by $k+1$, it is smallest if the remaining nodes are evenly distributed between all components.

Eventually, the span of the number of nodes takes its maximal value if there are only two components, one containing the minimal number of nodes possible and one with all other nodes. Since the smallest number of nodes in a component is $k+1$ the maximal span is $n-2(k+1)$. □

[4] Note that the span refers to the statistical property of the distribution of the number of nodes in the connected components.

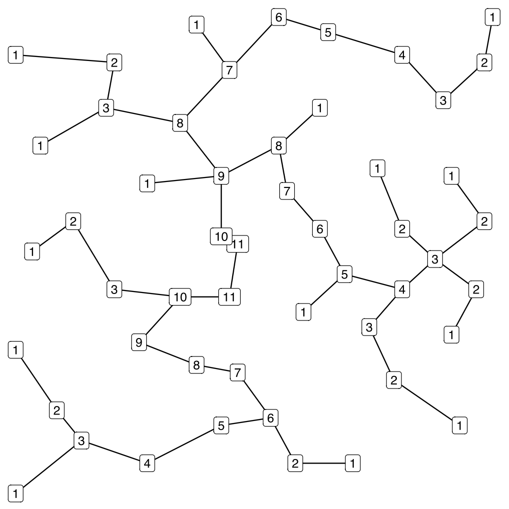

Fig. 3. MST node depth calculation on eil51.

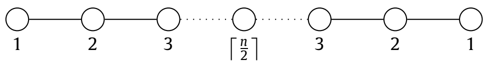

Fig. 4. An n-vertex path of odd length. Each node is labeled with its depth.

Theorem 2.5. *For the number of nodes in the strongly connected components of the k-NNG G_k^d of any Euclidean graph with $d : V \times V \to \mathbb{R}_{\geq 0}$ and $n - 2 > k$, the following holds:*

1. *The minimum and maximum values of the mean are $\frac{n}{n-k}$ and n, respectively.*
2. *The minimum and maximum values of the median are* 1 *and n, respectively.*
3. *The minimum and maximum values of the maximum are $k + 1$ and n, respectively.*
4. *The minimum and maximum values of the minimum are* 1 *and n, respectively.*
5. *The minimum and maximum values of the span are* 0 *and $n - 2$, respectively.*

Proof. Like in the case of weakly CCs (see Theorem 2.4) all statistics except for the span are maximal if there is only one single component containing all nodes. This is possible due to Theorem 2.1. As a consequence, the span of the number of nodes in the strongly CCs is minimal and zero in this case. As discussed in the proof of Theorem 2.3 the minimal possible number of nodes in a strongly connected component is one. Hence, one is the minimal possible value for the median if there are cases with more than half of the components having one node. This is satisfied if two components have one node and a third contains $n - 2$ nodes. The minimal mean value is reached if nodes are distributed between the maximum number of strongly connected components as described in Theorem 2.3. Therefore the maximum mean value is $\frac{n}{n-k}$. The maximal number of nodes in a strongly CC is minimal in the same case.

As argued in Theorem 2.3 there is always at least one component with $k + 1$ nodes; this is a lower bound for the maximal number of nodes in a single component.

The span is maximal if there are only two components, one containing the minimal number of nodes possible and one with all other nodes. Since the smallest number of nodes in a component is one the maximal span is $n - 2$. □

2.3. Minimum spanning tree depth features

Another crucial group of TSP features relies on a minimum spanning tree of the input instance.[5] Given an MST, this feature group consists of minimum, maximum, mean, median and the span of the node depth in an MST of the source graph. The algorithmic implementation of the node depth calculation is as follows: an initial depth value $D = 1$ is initialized. Next, the algorithm removes leaf nodes recursively labeling them with D and increasing D after each iteration by one. The process is repeated until all nodes are labeled. Fig. 3 visualizes the process on TSPLIB instance eil51.

Lemma 2.2. *Among all trees with n nodes, the n-vertex path has the maximal sum of depths.*

[5] Note that minimum spanning trees are not necessarily unique.

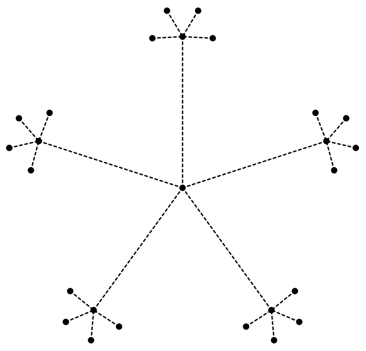

Fig. 5. The MST depicted by the dashed lines corresponding to the case of minimal mean depth.

Proof. Note that the depth calculation does an "onion-peeling" by iteratively removing leaf nodes. For a tree, in every iteration at least two leaf nodes are removed. Now on a n-vertex path the algorithm performs exactly $\lceil n/2 \rceil$ iterations while on any other graph the number of iterations is upper bounded by $\lceil n/2 \rceil$. □

Theorem 2.6. *For the n-vertex path we have:*

1. *The mean depth equals $\frac{n}{4} + \frac{1}{2}$ for even n and $\frac{\lceil n/2 \rceil^2}{n}$ for odd n.*
2. *The median depth is $\frac{n}{4} + \frac{1}{2}$ if $n \bmod 4 = 0$ and $\lceil \frac{n}{4} \rceil$ otherwise.*
3. *The maximum depth is $\lceil \frac{n}{2} \rceil$ and the span $\lceil \frac{n}{2} \rceil - 1$.*

Proof. Ad (1): We distinguish n even and n odd. For even n we see that each depth between 1 and $n/2$ is counted twice (see Fig. 4). Hence, the mean depth is given by

$$\frac{1}{n}\sum_{i=1}^{n/2} 2i = \frac{2}{n}\sum_{i=1}^{n/2} i = \frac{2}{n} \cdot \frac{(n/2)(n/2+1)}{2} = \frac{n}{4} + \frac{1}{2},$$

where we used the formula for the arithmetic sum $\sum_{k=1}^{n} k = n(n+1)/2$. For odd n we obtain

$$\frac{1}{n}\left(\sum_{i=1}^{\lfloor n/2 \rfloor} 2i + \left\lfloor \frac{n}{2} + 1 \right\rfloor\right) = \frac{\lfloor n/2 \rfloor(\lfloor n/2 \rfloor + 1) + (\lfloor n/2 \rfloor + 1)}{n}$$
$$= \frac{(\lfloor n/2 + 1 \rfloor)^2}{n} = \frac{\lceil n/2 \rceil^2}{n}.$$

Ad (2): Let $x_{(i)}$ denote the i-th order statistic of the depth values. Note that all but one depth value occur twice for an n-vertex path. Therefore, $x_{(\lceil n/2 \rceil)} = \lceil n/4 \rceil$. Hence, the median value is $\lceil n/4 \rceil$ for both odd and even n. The case where n is a multiple of 4 requires special attention. Here, the values $x_{(\lceil n/2 \rceil)}$ and $x_{(\lceil n/2 \rceil + 1)}$ differ by one. Therefore,

$$\frac{x_{(\lceil n/2 \rceil)} + x_{(\lceil n/2 \rceil + 1)}}{2} = \frac{n/4 + n/4 + 1}{2} = \frac{n}{4} + \frac{1}{2}.$$

Ad (3): The values for maximum depth and span are obvious. □

The structure of an MST with minimal depth is not trivial since a node within an MST of a Euclidean graph in the two dimensional space cannot have a degree greater than six as shown in [31]. In the following, we will first describe the construction of a graph with a variable number of nodes n that we will then show to have an MST of minimal possible depth.

In this regard, consider a graph $G = (V, E)$ that follows the fractal structure depicted in Fig. 5 and – "zoomed in" for more details – in Fig. 6. All nodes are placed around an *initial* node $I \in V$ with five nodes $P_i, 1 \leq i \leq 5$ being placed on the vertices of a regular *pentagon* and for all v_j $d(I, P_i) = r_1 < d(I, v_j),\ v_j \in V,\ v_j \neq P_i,\ 1 \leq i \leq 5$. Around the nodes P_i four nodes $C_{3,i,k},\ 1 \leq k \leq 4$ are placed. Note that the index of the nodes C_{lik} consists of three parts that locate the node in the fractal structure. The first part l states the level of the fractal structure the node belongs to, i identifies the node P_i that the considered node is allocated to and k counts the nodes with same l and i. For the four nodes around every P_i it holds that $d(C_{3,i,k},\ C_{3,i,l}) = d_3 > d(P_i, C_{3,i,l}) = r_2,\ 1 \leq k \leq 4,\ l \neq k,\ 1 \leq l \leq 4$. This ensures that $\forall l,\ \forall k,\ \forall f,\ \forall t\ d(C_{l,i,k}, C_{f,i+1,t}) < r_1$ where the radius r_2 is set to be $(d(P_i, P_{i+1}) - r_1)/3$. $C_{3,i,1}$ is placed to the left of P_i between the circle c_1 with radius r_1

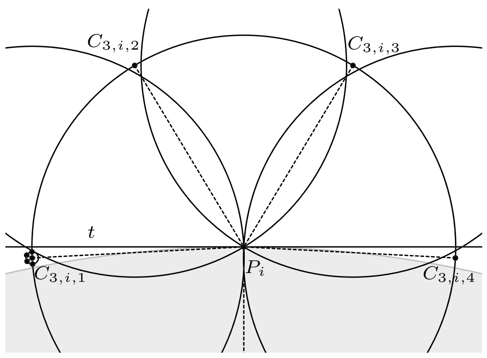

Fig. 6. The repetitive pattern if more nodes of further levels are added. Dashed lines are the MST-edges and circles represent the area within that no other node can be placed without a change of the edge of the MST. The grey circle part is from the circle around the initial node I. $C_{3,i,4}$ and $C_{3,i,1}$ lie between the border of this circle part and the tangent t at P_i. Around $C_{3,i,1}$ nodes of the next level are depicted.

around I and the tangent t of c_1 on P_i with a distance to t of one third of the distance between the intersections of t and c_1 with the circle c_2 with radius r_2 around P_i. The same applies for $C_{3,i,4}$ in a mirrored way. This construction pattern is repeated for nodes of greater levels in the fractal structure. Those nodes $C_{l,i,k}$ are added analogously but instead of I now $C_{l-2,i,\lceil \frac{k}{16} \rceil}$ is considered and instead of P_i now $C_{l-1,i,\lceil \frac{k}{4} \rceil}$ is considered. The MST $T = (V, E_T)$ of this graph contains the edges of G that connect the initial node with every P_i and those edges that connect every node with its four surrounding nodes of the next level. This is because G is constructed in such a way that those edges are the shortest of the node of the lower level to any other node.

Theorem 2.7. *The mean depth of an MST with $n > 7$ nodes is minimal if its structure corresponds to the structure of graph T.*

Proof. An MST with a smaller depth would either have more than five nodes around the initial node or on average more than four nodes around every succeeding node. As stated before the maximal number of nodes that can be connected to one single node within an MST of a Euclidean graph is six for the case of the two dimensional space [31]. If six nodes are placed around I the eighth point $C_{3,1,1}$ cannot lie between t and c_1 since the P-nodes have the same distance to I and to their neighboring P-nodes and following from that $d(P_5, I) > d(P_5, C_{3,1,1}) \vee d(P_2, I) > d(P_2, C_{3,1,1})$ and hence, $(I, P_5) \notin E_T \vee (I, P_2) \notin E_T$. Therefore, only two nodes of the next level can be connected to every P_i in this case. Thus, the MST of this case has a smaller depth for $n = 7$, an equal depth for $(n < 7) \vee (7 < n < 10)$ and a smaller depth for $n > 9$. If an MST would have on average more than four connected succeeding nodes around every node other than the initial node this means that some nodes would have five or six connected succeeding nodes. Six connected succeeding nodes are not possible since then no preceding node could be connected to the considered node. If a node is connected to five succeeding nodes the preceding node has to lie at the last free point for the case of six nodes around one node. Thus, the preceding nodes of the considered node could only have two preceding nodes as in the case for the six nodes around I yielding an average of less than four preceding nodes per node. □

With this, the minimal possible median depth of the MST is one since more than half of the nodes are leaves. The maximal depth in this case can be calculated by calculating the greatest completed level L of this tree starting at one as the depth. The level L is therefore the largest integer for which $1 + \sum_{i=0}^{L-2} 5 \cdot 4^i \leq n$. The number $\sum_{i=0}^{L-2} 4^i$ interpreted in the quaternary number representation is a string with all ones. This means multiplying this number with three and adding one yields a string in quaternary representation with a leading one and $L-2$ zeros, i.e. 4^{L-1}. Thus, $\sum_{i=0}^{L-2} 4^i = \frac{(4^{L-1}-1)}{3}$ and with this $4^{L-1} \leq \frac{3}{5}n + \frac{2}{5}$ and since L is the largest integer for which this inequality holds L can be computed by $L = \lfloor (\log_4 \left(\frac{3}{5}n + \frac{2}{5}\right) + 1 \rfloor$. With this, the number of remaining nodes R that are not sufficient to complete the current level can be calculated by $n - (\frac{5}{3} \cdot 4^{L-1} - \frac{2}{3})$. If $R > 4^{L-1}$ more than one branch accruing from the five P nodes has remaining nodes as its leafs. This means the depth of the root node and therefore the maximal depth is $L+1$ and, otherwise L. In both cases, the span is the max depth minus one. Second, the sum of all depths without R is computed by $L + \sum_{i=1}^{L-1} 5 \cdot 4^{i-1} \cdot (L-i)$. Then, the growth of the sum of all depths by adding R nodes can be calculated with a recursive function $f(r, l)$ with r being the number of nodes at level l that have a greater depth because of the remaining nodes R. If $l = 1$ the function evaluates to $r + \mathbb{1}_{r>1}$ where $\mathbb{1}_{r>1}$ is the indicator function which evaluates to 1 if the predicate $r > 1$ is true and to 0 otherwise, and else $r + f(\lceil \frac{r}{4} \rceil, l-1)$. Finally, the mean of all depths can be calculated by

$$\frac{1}{n}\left(L + \sum_{i=1}^{L-1} 5 \cdot 4^{i-1} \cdot (L-i) + f(R, L)\right).$$

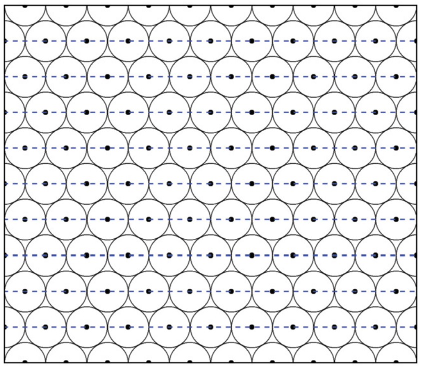

Fig. 7. Circles drawn around a sample instance with maximal possible minimal NN distance between all cities.

2.4. Calculation of d_{max}

A big part of the salesperson features is built upon the distance between two nodes. The feature groups concerned with those distances are the distance, the nearest neighbor distance and the MST-distances feature group. The last feature group is part of the MST-based feature group and the nearest neighbor distance features are closely related to the k-NNG features as they are statistics of the distances between nearest neighbors. Thus, both feature sets are promising features to extend the previously considered feature groups. Especially for those features the minimal distance between two nodes, i.e., the minimal nearest neighbor distance occurring in the instance, is of interest. While the lower bound of the minimal distance between two nodes is trivially 0 the upper bound needs to be derived with more thought. We will call this upper bounding distance d_{max} in the following and since it is the basis for different feature groups, we will cover the calculation of d_{max} in this section before we cover the different feature groups.

Consider the case that every distance from one city to its nearest neighbor is maximal and the same for every city. In this case a circle can be drawn around every city with a radius of $r_{max} = \frac{d_{max}}{2}$ such that none of those circles overlap. Only if this is the case no distance between two cities is smaller than d_{max}. Furthermore, the circles must be packed in the highest possible density so that r_{max} is the maximal possible radius around every city. This is known as the circle packing problem. For equal circles, Thue [35] proved that the most efficient packing is hexagonal packing, and thus, that the ratio of the area covered by the circles to the overall area is upper bounded by $\left(\frac{\pi}{2\sqrt{3}}\right)$ [35]. Hexagonal packing in this context means that every circle has six circles placed directly around itself except for those at the border of the bounding box. An example of this can be seen in Fig. 7.

With the formula for the area covered by the circles, the number of cities n and the formula for one circle area πr^2, the following equation can be derived for $n \to \infty$:

$$n\pi r_{max}^2 = \left(\frac{\pi}{2\sqrt{3}}\right) \cdot A. \tag{1}$$

Here, A denotes the area of the smallest box containing all cities (bounding box). However, this only holds true for $n \to \infty$. If for instance $n \leq 20$ circles should be fitted in a square the optimal solutions do not cover a $\frac{\pi}{2\sqrt{3}}$ fraction of the space and look structurally very different depending on n [26]. Thus, an exact calculation of d_{max} for every possible n and bounding box is not feasible. However, for large n the fraction of the space covered by the circles will approximate $\frac{\pi}{2\sqrt{3}}$ and we can therefore derive an upper bound that is not tight but close to the exact upper bound for large n.

Theorem 2.8. *For a TSP instance with $n \geq 2$ cities in a bounding box with side lengths a and b the following inequality holds:*

$$d_{max} \leq \frac{\frac{a+b}{2} + \sqrt{\left(\frac{a+b}{2}\right)^2 + 4(n-1)\left(\frac{ab}{2\sqrt{3}}\right)}}{n-1}.$$

Proof. Consider the case of regular circle packing: If for some n and a specific bounding box a packing is optimal that does not yield the highest possible density, Equation (1) would overestimate r_{max}. This is true since with the equation it would be assumed that more space of the bounding box is covered by the circles than they do. In the considered variation of circle packing, the circles can be partially outside of the considered bounding box if the centers of the circles, the cities, are still on the border of the bounding box. If many circles are partially outside of the bounding box, the maximal minimal distance between two cities can be greater since the circles at the border occupy less area of the bounding box. Thus, the minimal distance is maximal if the most possible cities are at the border of the bounding box and the remaining cities are packed in the densest possible manner. The most cities are at the border of the bounding box if one is at every corner of the bounding box and if cities are placed with a distance of d_{max} to each other on the edges. Hence, at most there are 4 cities with three quarters and $2(\frac{a}{2r_{max}} - 1 + \frac{b}{2r_{max}} - 1)$ cities with halves of their surrounding circles outside of the bounding box. The last calculation may not necessarily yield an integer value, but the calculated value will nevertheless be an upper bound for the cities on the edges. Further, the highest possible density of the internal packing is bounded from above by $\frac{\pi}{2\sqrt{3}}$. Therefore, by subtracting the sum of full circle areas that can maximally be outside of the bounding box from n in Equation (1) the following inequality can be derived:

$$\pi r_{max}^2 \left(n - \left(\frac{a}{2r_{max}} + \frac{b}{2r_{max}} + 1 \right) \right) \leq \frac{\pi}{2\sqrt{3}} ab.$$

Finally, by substituting $r_{max} = \frac{d_{max}}{2}$ and solving the inequality for d_{max} we get the inequality of the theorem. □

2.5. Nearest neighbor distance features

The nearest neighbor distance features are statistics on the distances from a node to its nearest neighbor. For the lower bounds it should be noted that not all distances can be 0, i.e., all nodes cannot lie at the same location, since the normalization is derived with respect to the bounding box of the instance which depends on the location of the nodes. Thus, if the edge lengths a and b of the bounding box are not 0 there has to be at least one nearest neighbor connection, whose edge length is not 0 as well.

Theorem 2.9. *For the nearest neighbor distances of an instance with n nodes, edge length a and b of the corresponding bounding box and a function $dm(n, a, b)$ that returns the upper bound for the maximal minimal distance d_{max} between nodes, the following holds:*

1. *The minimum value of the mean is* 0 *and the maximum value is smaller than* $dm(n, a, b)$.
2. *The minimum value of the median is* 0 *and the maximum value is smaller than* $dm(\lceil \frac{n}{2} \rceil + 2, a, b)$.
3. *The minimum and maximum values of the maximum are* 0 *and* $\sqrt{a^2 + b^2}$*, respectively.*
4. *The minimum and maximum values of the minimum are* 0 *and* d_{max}*, respectively.*
5. *The minimum and maximum values of the span are* 0 *and* $\sqrt{a^2 + b^2}$*, respectively.*

Proof. The mean nearest neighbor distance is minimal if all nodes lie at the same location. The nearest neighbor distance of all nodes is 0 in this case. Thus, the mean nearest distance is 0, too. Further, the mean is maximal if the sum of all nearest neighbor distances is maximal. This is equivalent to circle packing as described above with unequal sized circles and the goal to maximize the sum of the sums of the radii of the circles and their smallest neighboring circle. In the optimal structure, if a circle is not at the border of the bounding box it must touch at least three other circles since otherwise it could have a larger radius without the need to diminish the radius of another circle. Thus, on the one hand increasing a circle in such an arrangement that is bigger or equal to its neighboring circles will decrease the overall sum since the area the circle takes up grows quadratically with the radius which needs to be compensated by the remaining circles. A decrease of a circle that is bigger than its neighboring circles on the other hand will allow the neighboring circles to increase with a surplus in the growth of the overall sum. Hence, in the optimal case all circles have equal sizes and the optimal structure is hexagonal packing as in Section 2.4. The median nearest neighbor distance has a trivial lower bound. If more than half of the nodes lie at the same location the median is 0. The upper bound for the median is not the same as the one for the mean since almost the half of the nearest neighbor distances can be 0. Therefore, the half minus one nodes can lie at the same location and be treated as one node. Thus, the problem is finding d_{max} for $\lceil \frac{n}{2} \rceil + 2$ nodes.

The upper bound for the maximal nearest neighbor distance is the longest possible distance that can occur in an instance, namely $\sqrt{a^2 + b^2}$. This value is realized if all but one nodes lie at the same position and the last node spans the bounding box. The lower bound is achieved if all nearest neighbor distances are 0 as in the case of the minimal mean. The bounds for the minimal nearest neighbor distance are trivial.

The lower bound for the span is 0 which is the case when all nearest neighbor distances are the same as for the maximal mean. The upper bound equals the upper bound of the maximal nearest neighbor distance and is achieved in the same scenario. □

2.6. Minimum spanning tree distance features

The MST distance features are statistics on the edge lengths of the MST. Since the edge lengths of the minimal spanning tree depend on the shortest possible way to connect the cities to each other the upper bounds of the statistics depend on d_{max}.

Theorem 2.10. *For the edge lengths of an MST of an instance with n nodes, edge length a and b of the corresponding bounding box and a function $dm(n, a, b)$ that returns the upper bound for the maximal minimal distance d_{max} between nodes, the following holds:*

1. *The minimum value of the mean is* $\frac{\sqrt{a^2+b^2}}{n-1}$ *and the maximum value is smaller than* $dm(n, a, b)$*.*
2. *The minimum value of the median is* 0 *and the maximum value is smaller than* $dm(\lceil\frac{n}{2}\rceil + 2, a, b)$*.*
3. *The minimum and maximum values of the maximum are* $\frac{\sqrt{a^2+b^2}}{n-1}$ *and* $\sqrt{a^2+b^2}$*, respectively.*
4. *The minimum and maximum values of the minimum are* 0 *and* d_{max}*, respectively.*
5. *The minimum and maximum values of the span are* 0 *and* $\sqrt{a^2+b^2}$*, respectively.*

Proof. The minimal possible mean of the distances within an MST is obtained if the nodes are equally distributed across the shortest path that can define the bounding box. The nodes that define the path can lie on the corners of the bounding box in which case there only need to be two such nodes or at the edges of the bounding box requiring four such nodes. For the case of four nodes all nodes can be placed freely on their edge. On the edges with length b the fractions $q, p \in [0, 1]$ and on the edges with length a the fractions $u, v \in [0, 1]$ determine the location of the nodes. Assuming that q and u have a greater or equal value to their counterpart the shortest path length l can be calculated as: $l = \sqrt{((1-q)b)^2 + (va)^2} + \sqrt{((1-v)a)^2 + ((1-p)b)^2} + \sqrt{((1-u)a)^2 + (pb)^2}$. In order to minimize l the values u and q should be set to 1 which yields $l = va + \sqrt{((1-v)a)^2 + ((1-p)b)^2} + pb$. Finally, since squaring adding and then taking the square root of two values will be smaller than just adding them p and v should be set to 0. I.e., all four nodes are located on the two edges of the bounding box if l is to be minimized. Thus, the case with only two nodes defining the bounding box yields the shortest path. Therefore, the lower bound for the minimal mean of the distances within the MST is $\frac{\sqrt{a^2+b^2}}{n-1}$.

The upper bound is the same as for the mean of the nearest neighbor distance but not the argumentation. According to Prim's algorithm the MST contains all nearest neighbor connections. Thus, an MST can also be constructed by first, finding the undirected k-NNG with $k = 1$, second, creating a distance matrix for the distances between the components, where a distance between two components equals the smallest distance between any two nodes of the components and then repeating the process until there is only one component remaining. From Theorem 2.9 follows that the average nearest neighbor distances of the nodes cannot be greater than $dm(n, a, b)$. However, the nearest neighbor distances of the components correspond to second or higher nearest neighbor distances and could therefore increase the mean distance. To construct an MST with maximal distance the nodes in the components should be arranged equidistant as described in Theorem 2.9 for the maximal mean nearest neighbor distance. Consider in the following the case of two nodes per component, the argumentation, however, works with any number of nodes. Further components of components would also lead to the same result. If the two nodes of a component would lie at the exact same location the components can be viewed as individual nodes and therefore the sum of the nearest neighbor distances is again maximized with hexagonal packing. For $n \to \infty$ the d_{max} of this case is $\sqrt{2}$ times larger than the d_{max} of the base instance. However, since the distances of the two nodes within the components are 0, the mean of edge lengths within the TSP will still be smaller than d_{max}. If the distance between the nodes of the components is increased the shape that needs to be packed is that of two overlayed circles where one of them is shifted by the increase of the distance. This new shape needs more space and since the new shape gains the area of a sickle the additional space needed will be greater with the first increases of the distance between the two nodes than with last increases. Hence, since more and more increases lead finally to hexagonal packing with no components this means if there is a higher upper bound it must be a small variation of hexagonal packing. However, increasing the distance of one node to its neighboring nodes in hexagonal packing will cause at least one of the distances to the nearest neighbor of one of those neighboring nodes to decrease by the same amount. Thus, the maximal mean distance within an MST is d_{max}. The argumentation for the remaining statistics is the same as for Theorem 2.9. □

After deriving the theory of normalizing TSP features, we proceed with an empirical analysis and testing of our outcome in the following sections. With this we demonstrate the advantages of normalized TSP features.

3. Instances and algorithms

In this section, a brief overview of the used TSP instances, considered TSP solvers and validation of the solvers' runtimes is given. The full set of considered instances is the union of three sets from the literature enhanced by new, generated instances.

The first set was already used in Kerschke at al. [17] and is referred to as ECJ2018 in the following. It consists of a collection of instances from different sources, including artificially generated instances such as random uniform Euclidean (RUE) instances, strongly clustered instances (Netgen), and hybrids "in-between" resulting from "morphing" each one RUE and Netgen instance (Morphed); for details on the morphing process we refer the interested reader to [22]. In addition, the set contains real-world instances from the well-known TSPLIB [29] benchmark (TSPLIB), instances from the very large scale integration (VLSI) application area and instances considering subsets of real-world cities (National). Fig. 8 gives an exemplary impression of the structural differences between the instance types. In total, there are 1 844 TSP instances in this set. Next, we also used the instance set that was proposed in [5] consisting of 450 instances in total. The number of cities for each instance is $n \in \{500, 1\,000, 2\,000\}$. These instances were generated in an evolutionary process using sophisticated mutation operators with the goal to evolve instances which show severe performance differences between state-of-the-art solvers EAX and LKH (see below). We refer to this set as FOGA2019. The third set of instances is taken from [33]. It consists of

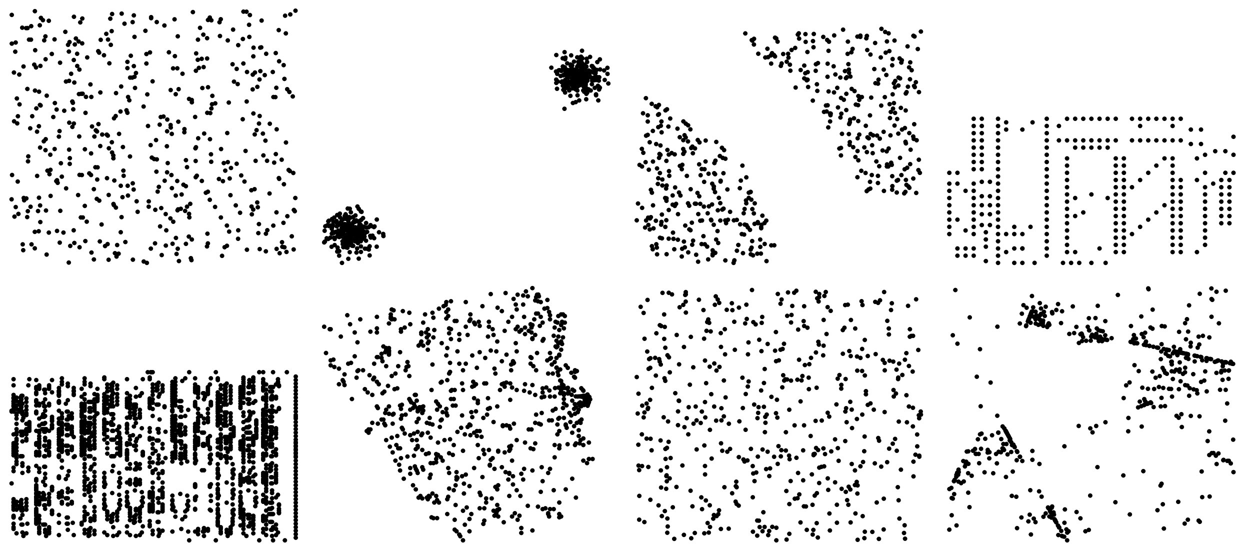

Fig. 8. Exemplary instances from the considered instance set. From left to right row by row: RUE, Netgen (two dense clusters), Morphed (combination of the two preceding instances), TSPLIB, VLSI, National, and evolved (with simple mutation) as well as evolved (with sophisticated mutation).

Table 1
Overview of all instance subsets including the PAR10 and PQR10 scores for the three considered solvers EAX, LKH and Concorde (best performing solvers are highlighted in **bold** while second bests are underlined).

Set	Num. of Cities				Num. of Instances	PAR10			PQR10		
	Min.	Median	Mean	Max.		EAX	LKH	Concorde	EAX	LKH	Concorde
					ECJ2018 [17]						
RUE	500	1 250	1 250	2 000	600	**16.53**	163.13	1 319.92	**12.82**	150.08	1 284.95
VLSI	662	1 412	1 311	1 973	18	**5.46**	59.85	6 544.21	**4.68**	41.91	6 230.43
TSPLIB	574	1 291	1 195	1 889	21	**1 726.82**	1 794.12	11 554.98	**1 722.84**	1 764.66	11 849.54
National	634	866	1 028	1 979	5	**4.59**	19.63	513.60	**3.55**	12.41	471.38
Netgen	500	1 250	1 250	2 000	600	**10.16**	669.33	253.84	**8.49**	574.24	248.21
Morphed	500	1 250	1 250	2 000	600	**13.88**	678.12	481.73	**10.70**	583.05	466.47
					FOGA2019 [5]						
All	500	1 000	1 167	2 000	450	**10.89**	421.90	798.92	**8.61**	410.19	779.91
					Evolved-Simple Instances						
500-EAX	500	500	500	500	1 000	**1.25**	5 524.22	43.37	**1.12**	5 127.05	42.22
500-LKH	500	500	500	500	1 000	110.23	**0.89**	62.39	80.71	**0.73**	60.97
1 000-EAX	1 000	1 000	1 000	1 000	500	**4.79**	3 940.62	364.46	**3.98**	3 306.38	357.22
1 000-LKH	1 000	1 000	1 000	1 000	500	146.28	**44.29**	385.05	101.38	**44.04**	376.17
1 500-EAX	1 500	1 500	1 500	1 500	500	**12.65**	2 818.57	706.16	**10.14**	2 504.68	686.15
1 500-LKH	1 500	1 500	1 500	1 500	500	208.57	**14.70**	667.13	131.27	**10.79**	651.24
2 000-EAX	2 000	2 000	2 000	2 000	500	**25.35**	2 877.67	1 399.64	**19.62**	2 458.15	1 367.76
2 000-LKH	2 000	2 000	2 000	2 000	500	258.61	**42.05**	1 263.46	155.46	**25.44**	1 233.53
					Evolved-Sophisticated Instances						
500-EAX	500	500	500	500	1 000	**1.01**	7 987.63	23.15	**0.95**	7 952.74	22.42
500-LKH	500	500	500	500	1 000	62.17	**1.32**	49.80	46.53	**1.08**	48.67
1 000-EAX [33]	1 000	1 000	1 000	1 000	500	**3.92**	7 128.84	207.23	**3.40**	6 994.79	202.97
1 000-LKH [33]	1 000	1 000	1 000	1 000	500	138.25	**7.73**	281.47	88.75	**5.49**	273.51
1 500-EAX	1 500	1 500	1 500	1 500	500	**9.49**	6 192.82	534.80	**7.79**	5 422.83	516.74
1 500-LKH	1 500	1 500	1 500	1 500	500	286.27	**16.54**	615.87	165.50	**12.91**	597.14
2 000-EAX	2 000	2 000	2 000	2 000	500	**19.17**	4 480.48	1 196.17	**15.25**	4 064.28	1 161.64
2 000-LKH	2 000	2 000	2 000	2 000	500	225.05	**38.14**	1 119.17	135.98	**22.73**	1 091.05

1 000 instances with a size of $n = 1\,000$ cities evolved by the method proposed in [5]. 500 of these instances are easy to solve for EAX and the other 500 are easy to solve for LKH. In addition, we generated another large set of evolved instances with $n \in \{500, 1\,500, 2\,000\}$. In combination with the instances from [33], this set is referred to as evolved instances. We extended the set of [33] with additional evolved instances to increase the diversity of the artificially generated instance set. Hence, we considered not only different numbers of cities but also the simple mutation operators [5] which have noticeable different characteristics in comparison to the sophisticated mutation operators. For an overview of all considered instance sets see Table 1 and Fig. 8 for some visualizations of exemplary considered instances.

We are interested in improving the state of the art in TSP-solving by means of per-instance algorithm selection. Therefore, our algorithm portfolio consists of the following three algorithms: *edge assembly crossover* (EAX) [23], *Lin-Kernighan-Helsgaun heuristic* (LKH) [12,20], and the exact solver Concorde [1]. EAX is a genetic algorithm with a smart crossover operator (termed EAX as well) and entropy-based diversity preservation to prevent premature convergence of the population. LKH is Keld Helsgaun's implementation of a highly effective local-search algorithm based on k-OPT edge exchanges. Both solvers, in their respective restart versions [10], pose the state-of-the-art in inexact TSP-solving [17,19].[6] Concorde on the other hand is an exact solver based on a highly developed ILP-based branch-and-cut procedure involving cutting-plane methods; Concorde is frequently capable of solving instances with thousands of nodes to optimality in reasonable time which is astonishing given the problems' NP-hardness.

We solve all TSP instances and benchmark their performance against each other. Each solver was executed ten times on each instance and its runtime was recorded. We then calculated the *penalized average runtime* (PAR10; [2]) and the less outlier-sensitive *penalized quantile runtime* (PQR10; [15]) scores for each solver and instance. Both measures aggregate the runtime over the ten runs. When a solver was not able to find the optimum within the give time-budget, the solver receives ten times the cutoff-time as penalty. In contrast to PAR10, which computes the arithmetic mean of the penalized runtimes, PQR10 is based on the p-quantile, with $p = 0.5$ (i.e., the median) in our case. We used a cutoff-time of 3 600 seconds for EAX and LKH while Concorde had an unlimited runtime. The reason is that EAX and LKH are heuristic solvers. Both do not guarantee to find the optimal solution to a given TSP instance. Instead, they may get stuck in a local optimum and even an unlimited time-budget will not prevent that. In contrast to EAX and LKH, Concorde is an exact solver — it will find the optimum within an unlimited period of time. The PAR10 and PQR10 scores for the different solvers and instance sets can be found in Table 1. It becomes evident, that for all instance sets, EAX and LKH can be considered to be competitive to each other. In fact, 51.6% of all instances are EAX-easy instances — meaning, that EAX is the quickest solver to find the optimum. In addition, 48.3% of all instances are LKH-easy while only 0.04% of all instances are Concorde-easy. We found that Concorde is not sufficiently competitive to EAX and LKH and is, therefore, not considered in our case study.

4. Exploratory data analysis

Next, we conduct an exploratory data analysis (EDA) on the instance groups introduced in Section 3 to assess the normalization effects of the MST, k-NNG, and nearest neighbor features visually. First, we evaluate how the normalization affects the TSP features themselves. Then, we deepen the analysis by assessing how the normalization affects the correlation of the features (a) to other features and (b) to the performances of EAX and LKH.

4.1. Normalized vs. unnormalized features

We first compare scatterplots of normalized and unnormalized features, respectively. Some exemplary yet representative results on the FOGA2019 instance set can be found in Fig. 9. Since we already presented the effect of the normalization on some features in our previous study, we will consider the newly added features in this plot. Hence, the maximal distance of a node to its nearest neighbor is shown on the x-axis. On the y-axis, the mean edge length of an MST is plotted. The plot on the left shows the unnormalized features, the one on the right the respective normalized counterparts.

Considering the node coloring in the unnormalized plot, the different instance sizes ranging from 500 to 2 000 nodes are distinguishable. We observe that the mean distance in an MST increases strongly with the instance size, while the maximal distance of a node to its nearest neighbor only scales slightly with the instance's size. Thus, the clusters in the plot representing the instance size are clearly distinguishable. In contrast, in the normalized plot, the instance sizes are not the discriminating factor. The distinctively colored clusters representing different instance sizes overlap clearly, especially when examining the y-axis. Considering the other instance groups – ECJ2018 and evolved instances – the respective scatterplots show the same pattern. These results support the expected effects of the normalization. Concededly, when considering the elementary normalization by Pihera & Musliu [27], the results are visually similar. Nevertheless, the normalization conducted here is much more precise and mathematically well-founded. It, therefore, bears more valuable theoretical insights into the effects of TSP feature normalization.

Admittedly, a few other plots of normalized features still exist in which the instance size can still be distinguished. Here, the instances with a higher number of nodes have lower values than instances with fewer nodes. Nevertheless, as stated in the introduction of Section 2 this phenomenon has a reasonable mathematical explanation: our normalization considers the lower and upper bounds of the feature values, $f_{\min}$ and $f_{\max}$. For some features, e.g., the maximal depth within an MST, the upper bound grows linearly while the lower bound grows logarithmically with n. Thus, if the feature value on average grows logarithmically with n, the normalized value approaches zero for increasing n.

Furthermore, we explored the effect of the normalization when considering multiple features. To this end, we adopt autoencoders, a modern dimensionality reduction method. Autoencoders are feed-forward deep neural networks that learn data representations unsupervised. Autoencoders are composed of encoder and decoder networks and a latent space in-between. The encoder uses the usually high-dimensional input data and transforms it into a low-dimensional representation

[6] The restart-version triggers a restart once the algorithm runs into its internal stopping conditions and the given time budget is not yet depleted.

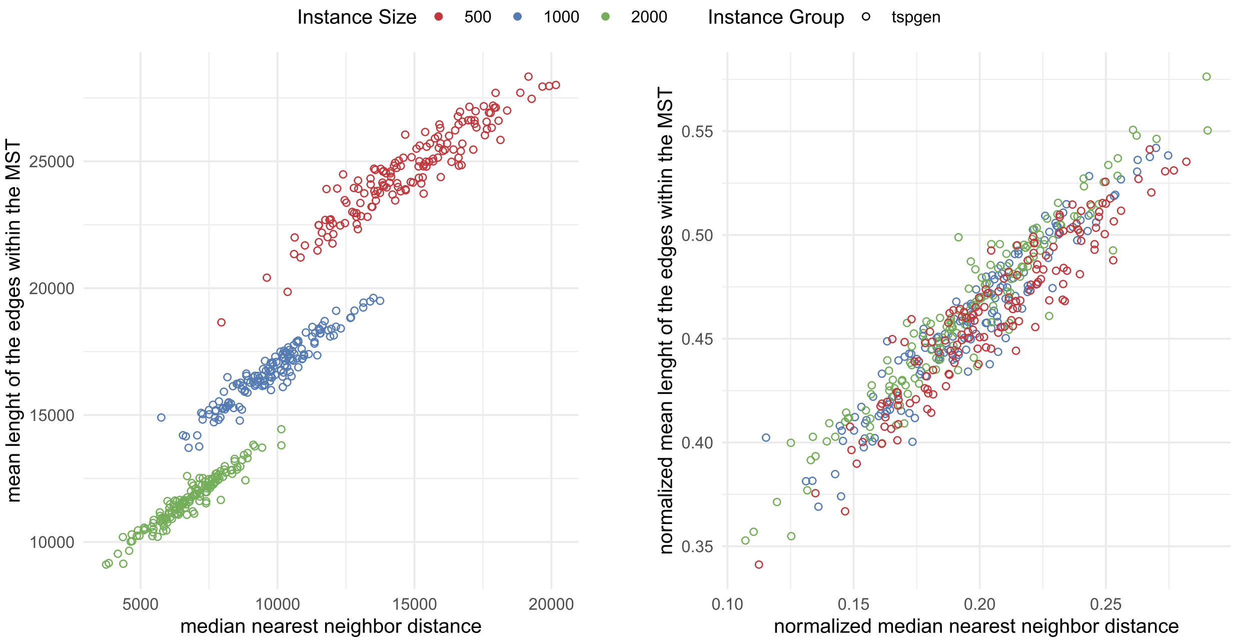

Fig. 9. Scatterplots of two 3-NNG graph features displaying the effect of the TSP feature normalization based on the FOGA2019 instance set. On the left-hand side, the unnormalized features can be seen, and on the right-hand side their normalized counterparts. The normalization worked as expected since the instance sizes cannot be distinguished clearly anymore. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

Table 2
The aggregated trustworthiness and continuity scores for the z-closest neighbors of the 30 independently trained autoencoders. Performing a one-sided Wilcoxon test yields that the autoencoder on normalized features have a significantly higher continuity and trustworthiness (p-value $\ll 0.05$).

	Unnormalized				Normalized			
	trustworthiness		continuity		trustworthiness		continuity	
z	median	iqr	median	iqr	median	iqr	median	iqr
1	0.815	0.008	0.936	0.012	0.956	0.012	0.992	0.004
5	0.814	0.008	0.917	0.011	0.955	0.012	0.989	0.005
12	0.813	0.009	0.900	0.009	0.954	0.013	0.987	0.007

in the latent space. The decoder then uses these low-dimensional encodings to retrieve the original high-dimensional data. Overall, the goal is to minimize the autoencoder's reconstruction error to recover an exact low-dimensional representation of the data. The effectiveness of autoencoders was first shown in [13]. We also considered other prominent dimensionality reduction methods like principal component analysis (PCA), multidimensional scaling (MDS), and t-distributed stochastic neighbor embedding (t-SNE). However, we decided against these classical approaches since they have problems capturing local relationships as was shown in [32] and/or perform worse than autoencoder (see, e.g., [13]).

The autoencoders were initialized with random weights. The resulting mappings vary between autoencoders, even though they were trained using the same data set. Hence, we trained 30 networks for the normalized data and 30 for the unnormalized data and found very similar performance and revealed the same pattern. The performance was assessed using trustworthiness and continuity as proposed in [36]. *Trustworthiness* measures to what extent the z-closest neighbors in the embedded space are also close in the original space, and *continuity* to what extent the z-closest neighbors in the original space are also close in the embedded space. Hence, trustworthiness penalizes close neighbors in the embedded space that are not close in the original space, and continuity penalizes neighbors in the original space that are not close in the embedded data. Both values range from 0 to 1, with values closer to 1 being better than values closer to 0. The results of our experiments can be seen in Table 2 for both the unnormalized (left part of the table) and the normalized features (right part). Although it is clearly visible that autoencoders encode the normalized features better, we performed a one-sided Wilcoxon test.

The test yields that both trustworthiness and continuity scores are significantly higher for the autoencoders based on the normalized features (compared to those based on the unnormalized features). It is thus evident that the normalization could substantially improve the continuity and trustworthiness of the autoencoders. This implies that autoencoders based on the normalized features can preserve the original local structure more than those based on the unnormalized features. Fig. 10 shows the result of a representative autoencoder mapping. It reduced the set of 49 nearest neighbor, NNG-, and

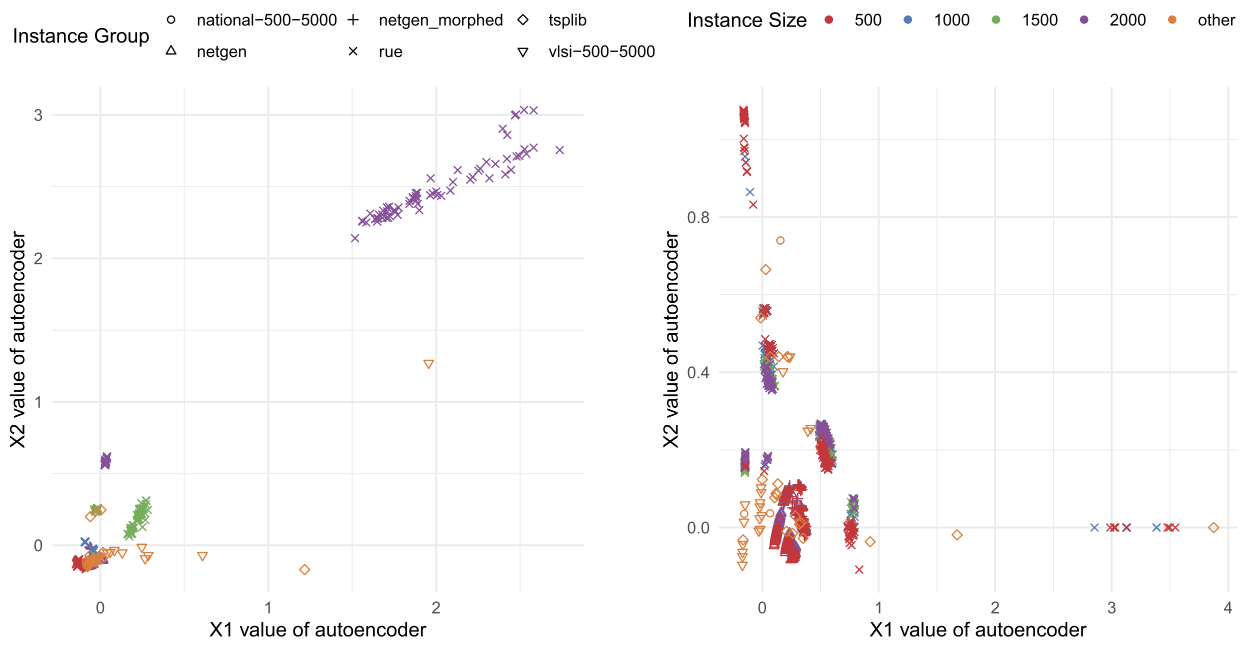

Fig. 10. Result of the reduction of 40 nearest neighbor, MST- and NNG-features to the two-dimensional space by using autoencoders. The plots show the two-dimensional representations for the unnormalized features (left) and the normalized features (right).

MST-features to the two-dimensional space represented by X_1 and X_2. Note that the underlying autoencoders have been trained on all instance sets listed in Table 2. However, we only visualized mappings for an exemplary subset of the instances for better readability, namely ECJ2018.

In the unnormalized plot, the different instance sizes can be discriminated clearly, indicated by the coloring of the instances. In the normalized plot, the instances of different sizes are not separable anymore. This is in line with our observations on feature pairs (see Fig. 9) and indicates that the normalization has the desired effect.

Next, we analyzed the correlation of the features to each other. Heatmaps of the correlation coefficient for the normalized and unnormalized features on the FOGA2019 set can be seen in Fig. 11. High positive correlations are colored in red, whereas blue color tones indicate a negative correlation.

As shown in the upper plot, the correlation is diverse among the set of features. Noticeably, there are light areas in the unnormalized feature correlation plot, which indicate a correlation close to zero. Nevertheless, there exist groups of highly positively correlated features. In particular, a red block in the middle of the plot, consisting of NNG-features related to the minimal, maximal, and span features of the strongly and weakly CCs, stands out. These features are also positively correlated with the features related to the depth of the respective MST.

In contrast to that, when considering the normalized heatmap in Fig. 11, the overall impression of the correlation changed notably. Here, the groups of highly correlated features are significantly smaller. The large block of highly correlated features is not visible anymore. The normalization revealed that the correlation between these features is not as high as it seemed. Additionally, when considering the relation of these features to the MST features, the correlation flipped in the negative space. A similar phenomenon can be observed when considering the features calculated on the evolved instance set. Also, for the ECJ2018 instances, the correlation decreased or shifted notably in the negatively correlated space, although a large group of positively correlated features remains. When considering these findings, eventually, researchers might have overestimated the intensity of the feature's interrelation in the past. This insight is interesting concerning the algorithm selection study, which we will conduct in Section 5.

4.2. The impact of feature normalization on the solver performance

Finally, the relationship of the solvers to the features is investigated by calculating the correlation of all normalized and unnormalized features to the performance of EAX and LKH. Notice that in Fig. 12, the points depict unnormalized or normalized features and not the different instances anymore. The plot reflects the correlation of the features with the PAR10 value of LKH on the abscissa (x-axis) and the PAR10 value of EAX on the ordinate (y-axis). This plot was created based on the evolved instances. Notice that in the normalized feature's plot, we plotted arrows indicating the shift from the former position in the unnormalized plot for a more straightforward interpretation.

Ultimately, the overall goal of the feature normalization is to gain insights for an algorithm selection study as we will conduct in Section 5. Having this in mind, the points in Fig. 12 representing the features should ideally reside in the second (top left) or fourth quadrant (bottom right) of the plot. Here, the correlation of the features to the performance of one

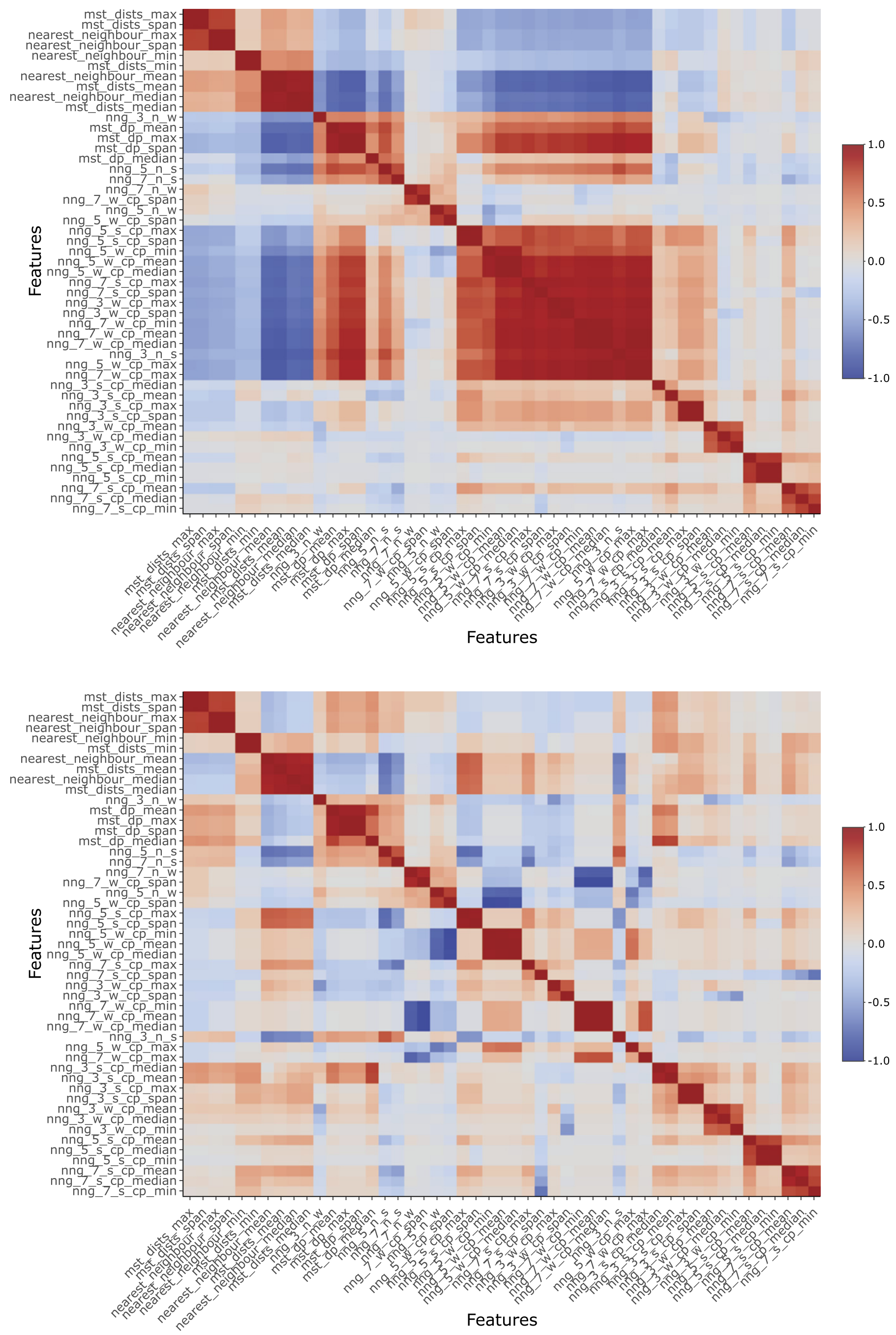

Fig. 11. Feature correlation heatmaps (top: unnormalized, bottom: normalized).

algorithm is positive and negative to the other algorithm. These features may facilitate the algorithm selection process. To better visually separate the favorable from the unfavorable quadrants in Fig. 12, we colored the favorable ones in blue and the less favorable ones in red.

In the left part of Fig. 12 – where the correlation of the unnormalized features to the solver's performance can be seen – only 25% of the features are located in one of the favored quadrants. Nearly half of the features reside in the upper right quadrant, indicating a positive correlation of the features to the performance of both algorithms. Interestingly, the features with the lowest correlation with EAX and LKH are those recently incorporated in this study.

In contrast, when considering the normalized features, ambivalent changes for different feature groups can be seen: for some features, the solver performance's correlation with the features decreased, for other features, the normalization has

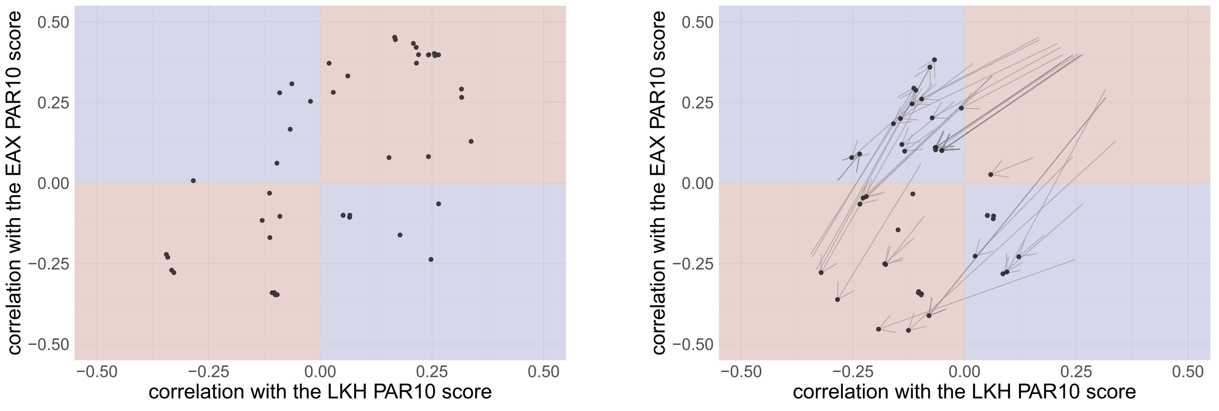

Fig. 12. Correlation of LKH and EAX performances with the features. The two plots show the correlations of the algorithms' performances with the unnormalized features (left) and normalized features (right), respectively. Additionally, in the normalized feature plot, the arrows indicate the shift of the features that is caused by their normalization.

no effect, while it increased for the newly added features. The same result can be observed when considering the ECJ2018 or FOGA2019 instance sets, respectively.

For the features that previously resided in the upper right quadrant, the correlation of the features to the algorithm performance shifted in the uncorrelated or negatively correlated space. This result is in line with the observation that the overall correlation of the features depicted in Fig. 11 declined. The features related to the span, the mean, and the maximal value of the depth of the MST moved furthest, from the positively correlated space in the left part of Fig. 12 to the negatively correlated space in the right part of Fig. 12.

When considering the arrows' flows belonging to this feature group, we observe that for some features, the correlation to the performance of LKH decreased more, and for other features, the correlation to the performance of EAX. Thus, the normalization generates an overall arched movement of the arrows. However, this phenomenon is only observable for the evolved instances and neither for the ECJ2018 nor FOGA2019 instance sets. For the last two instance sets, the arrows are arranged straight and indicate no difference in the decrease of the correlation for one of the algorithms. The evolved instances created by [5] and [33] were generated to be solved more easily by either EAX or LKH. The normalization reveals that this goal was achieved since most of the features calculated based on these instances correlate more with one algorithm.

Unaffected by the normalization is the correlation of ten features, which can be seen in the middle of both plots. Three of them are the minimal number of nodes in a strongly CC in a 5-NNG, the span of the number of weakly connected components in a 5-NNG, and the span of the number of weakly components in a 7-NNG. That is the case since many unstructured instances consist of a single weak component per definition. Similarly, the minimal number of nodes in a strongly CC in a 5-NNG equals one. Therefore, the correlation did not change with the normalization for these three features. The feature values are constantly zero or one, respectively. Additionally, the maximal number and the span of the number of nearest neighbors nodes and maximal distances in an MST, belong to this set. This is the case for those features as their theoretical upper and lower bound do not depend on n but only on the instance size.

For most of the newly added features, i.e., the nearest neighbor and the MST distance features, the correlation instead increased, especially when considering the correlation with EAX on the y-axis. The mean and median distances in an MST and the mean and median number of nearest neighbors, moved furthest from the lower left to the upper left quadrant. Fortunately, they moved from the red to the blue quadrant so that now, most of the features are located in the two blue quadrants. This result is a desirable effect of the normalization since it indicates that some features are more suitable for distinguishing algorithm performance than initially thought. These effects are also observable for ECJ2018 or FOGA2019 instances. Notice also that a good portion of features – independent of their normalization status – are slightly more correlated with the performance of EAX than with the performance of LKH. This might also explain why AS models – e.g., the ones in Section 5 – tend to select EAX more often than LKH.

After conducting the EDA, we can draw a few conclusions: first, the normalization effects are visible. As can be seen in Fig. 9 and 10, the instance sizes are no longer the primary discriminating factor. Second, the correlation of the normalized features to other normalized features decreased notably in contrast to their unnormalized counterparts. Third, the correlation of the features to the performance of EAX and LKH changed notably. For some features, the correlation decreased notably, while for others, it increased. Generally, it is revealed that most features are positively correlated with the performance of one algorithm and negatively with the performance of the other algorithm. These features might be more auxiliary for an ML algorithm to predict the performance of both algorithms as estimated.

All instance sets, algorithm performances, and plots are available online in a dashboard for further review.[7] In the following, we will test the suitability and effects of the normalized features for constructing automated algorithm selection models.

5. Case study: algorithm selection

The exploratory data analysis (see Section 4) revealed some changes in terms of feature visualization, which were caused by our proposed normalization. In the following, we will thus explore the potential of the normalized features for automated algorithm selection. Note, however, that the objective of this work is not a comprehensive benchmark study. Instead, the goal of this section is a proof-of-concept study that may provide a first indication about the suitability of the proposed normalization.[8]

Within our study, we utilized the following (normalized) features:

- summary statistics (mean, median, maximum and span) of an instance's MST depth (see Section 2.3),
- summary statistics (minimum, mean, median, maximum and span) of the distances in an instance's MST (see Section 2.6),
- the number of weakly/strongly connected components in the k-NNG of a TSP instance (see Section 2.2),
- summary statistics (minimum, mean, median, maximum and span) of the number of cities across the weakly/strongly connected components of an instance's k-NNG (see Section 2.2), as well as
- summary statistics (minimum, mean, median, maximum and span) of the distances between nearest neighbors in an instance (see Section 2.5).

The NNG-features were based on k-NNGs with $k = \{3, 5, 7\}$ following the parameterization proposed in [27].[9] Hence, our data contained a total of 50 features. The summary statistics in the set of unnormalized features are further enhanced with higher moment summary statistics – i.e., the standard variation, the coefficient of variation and the skewness – as they are part of the original feature sets. We then had to discard the normalized (unnormalized) minimum number of cities in the strongly connected component of a 3-NNG (feature `nng_3_strong_components_min`) from our data, as it was constantly zero (one) across all considered instances and thus did not provide any information for our candidate selectors. Instead, we occasionally also considered the instance size as a feature as it might nonetheless be useful for discriminating the different TSP instances.

In line with previous AS studies in the context of (inexact) TSP solving, our experiments are based on the 1 844 ECJ 2018 instances [17] (see Table 1 for details). We further focused on the two state-of-the-art heuristic TSP solvers EAX [24] and LKH [12] – more precisely, on their respective restart-variants [10]. As candidates for the AS models, we investigated random forests, support vector machines and gradient boosting, all of which are known to be very powerful machine learning algorithms. To avoid undesirable side effects such as redundancy and partially high correlations among features (see Section 4) and/or noise within features, we also performed automated feature selection. For simplicity, and in line with previous AS studies, we considered a greedy forward-backward selection strategy, which usually tends to find small but informative feature subsets in a very efficient manner. Note that we also considered the overhead of feature calculation. Yet, the costs for the considered (normalized) feature sets, i.e., the runtime for computing them, are negligible.

In terms of the selectors' performance assessment, we used the PAR10 score [2]. Moreover, further extending the previously published conference version of this work, we repeated our experiments using the PQR10 score [6] – a robust alternative to the widely used PAR10 score. For a simpler comparison of the performance values, it is recommended to compare the performance of the AS model with (a) the performance of the best standalone optimization algorithm – also known as *single-best solver* or SBS (here: EAX) – and (b) the best possible performance, which corresponds to an oracle-like (feature free) prediction of the best optimization algorithm per instance (also known as *virtual-best solver* or VBS). To facilitate this comparison, we subsequently converted the selector's PAR10 (PQR10) score into a closed-gap-percentage, indicating the proportion of the gap between the SBS and VBS performances, which is closed by the AS model. Note that this is a common metric in various AS studies [2,16,21].

In order to also take the stochasticity of the AS models into account, we repeated all our experiments 25 times. The resulting closed-gap-values are illustrated in Fig. 13 by means of boxplots. Note that xgboost is deterministic, and therefore, no variability in its performance can be observed across the 25 runs. As can be seen, the boxplots that correspond to the AS performances of the normalized features tend to be better with respect to the median closed gap than the ones of the unnormalized features. This observation is confirmed by Wilcoxon-Mann-Whitney tests at a significance level of $\alpha = 0.05$

[7] The dashboard can be found at https://tsp-features.shinyapps.io/normalization-eda/ and the corresponding implementation at https://github.com/jonathan-h1/TSP-EDA.

[8] The code for the experiments can be found at https://github.com/jonathan-h1/FOGA21_TSP_Experiments.

[9] In their work, Pihera & Musliu also considered values of k which depend on n, e.g., $k = \sqrt{n}$. We neglect these versions since the feature values for greater k-values do not differ from those attained with smaller k-values in most cases. The reason is that the number of weakly/strongly CCs in the k-NNG is monotonically decreasing as k approaches n. Hence, there is a $0 < k' < n$, such that for all $k \geq k'$ the number of connected components in the k-NNG is one.

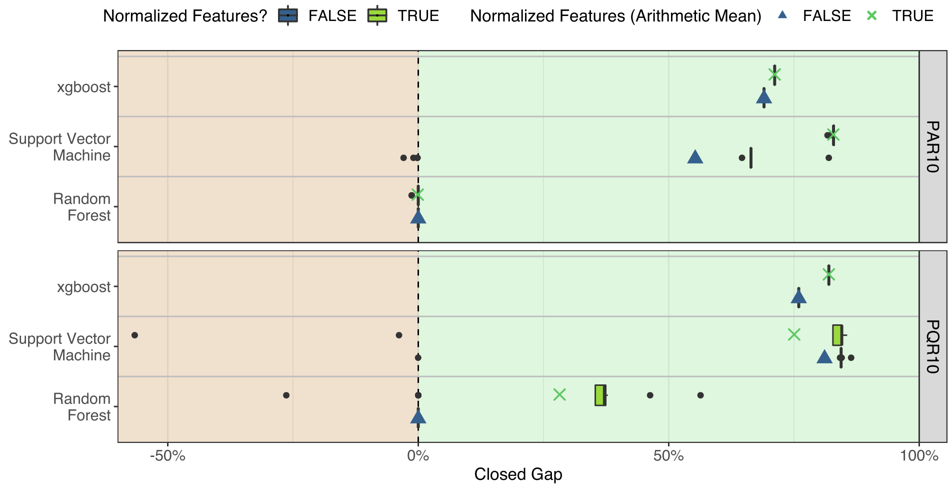

Fig. 13. Comparison of the closed-gap-performance of different AS models, which are based on different performance scores: PAR10 scores (top panel) and PQR10 scores (bottom panel). Each row visualizes the distribution of selector performances for a given model (row name) based on normalized (top) and unnormalized (bottom) features. Moreover, for each of the trained models, the arithmetic mean of the corresponding closed-gap-performances is depicted as blue triangle (unnormalized features) or green cross (normalized features), respectively.

Table 3
Misclassification counts and costs of xgboost. Results are shown for the selector based on the unnormalized and normalized features and distinguished by pair of true (rows) vs. predicted algorithm (columns). If the selector makes the correct prediction (EAX-EAX and LKH-LKH) the costs indicate the feature costs.

	Misclassification counts				Misclassification costs			
	unnormalized		normalized		unnormalized		normalized	
	EAX	LKH	EAX	LKH	EAX	LKH	EAX	LKH
EAX	986	162	1 017	131	0.26	32.97	0.04	22.36
LKH	454	242	537	159	9.13	0.10	8.05	0.02

in all cases except for random forests tested on PAR10 scores. Still, in most cases, there appears to be quite large variability across the performances. However, random forest models in the case of unnormalized features are degenerated at a 0%-level (see Fig. 13). This can be attributed to the models predicting EAX in most cases for instances that are actually easier for LKH. We stress that the results are particularly positive if one takes into consideration the severely reduced number of features used in this study.

Due to the high volatility of the performances, we examine the most promising models in more detail below. At first, we take a closer look at the results of xgboost in the PQR10 setting. Due to the nonstochasticity the same feature set is selected during every replication of xgboost. In the case of unnormalized features the selected ones are the span of the number of nodes in the weak and the median and minimal number of nodes in the strong CCs of the 5-NNG, the minimal number of nodes in the weak and the variance coefficient of the number of nodes in the strong CCs of the 3-NNG, as well as the skew of the depth values within the MST. In the normalized case the selected features are the span of the number of nodes in the weak CCs of the 7- and 5-NNG and the median number of nodes in the strong CCs of the 5-NNG. The selector based on the unnormalized features used more features than its counterpart, which has been modeled using the normalized features. The latter model is less complex (and thus more robust) resulting in lower overall feature costs (see Table 3). In tendency, the model based on the normalized features favors EAX over LKH comparably stronger than the model based on unnormalized features (see Table 3). Nevertheless, the model based on normalized features is more promising as it is able to decrease the misclassification costs in all categories (see Table 3).

Complementing the previous investigations, we also examined the results of the support vector machine in case of the PAR10 score setting. Fig. 14 visualizes, which features have been selected by the support vector machine in each of the 25 replications. Similarly to xgboost, the selectors required in tendency less features if they were normalized. However, the models used a more versatile set of features and included MST-distance and nearest neighbor distance features as well. Note that in case of the unnormalized features, two experiments (replications 13 and 20) crashed and are not considered further. Contrarily to xgboost, the models based on the unnormalized features tend to predict EAX more often; in fact, three models (replications 4, 11 and 16) almost always predict EAX (see Fig. 15). Nevertheless, the models based on the normalized features can reduce the misclassification cost of predicting LKH, if EAX is the ground truth, significantly (see Fig. 16). There is only one exception (replication 8), during which other features were selected, causing a strong bias towards selecting EAX. Noticeably, the selector of that particular replication is the only one, which (a) did not consider `mst_dists-sum` as one of its relevant features, and (b) used `nearest_neighbour-min` as one of its features.

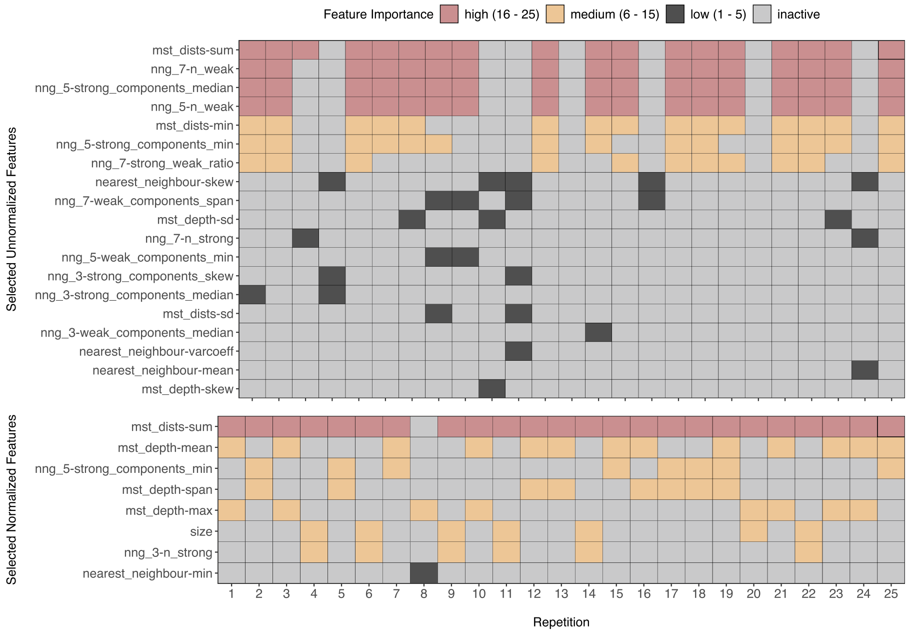

Fig. 14. Features (rows) selected by the support vector machine in the 25 replications (columns). The trained models are based on the unnormalized features (top), as well as the normalized features (bottom). The cell colors indicate how often a feature has been selected across all 25 replications.

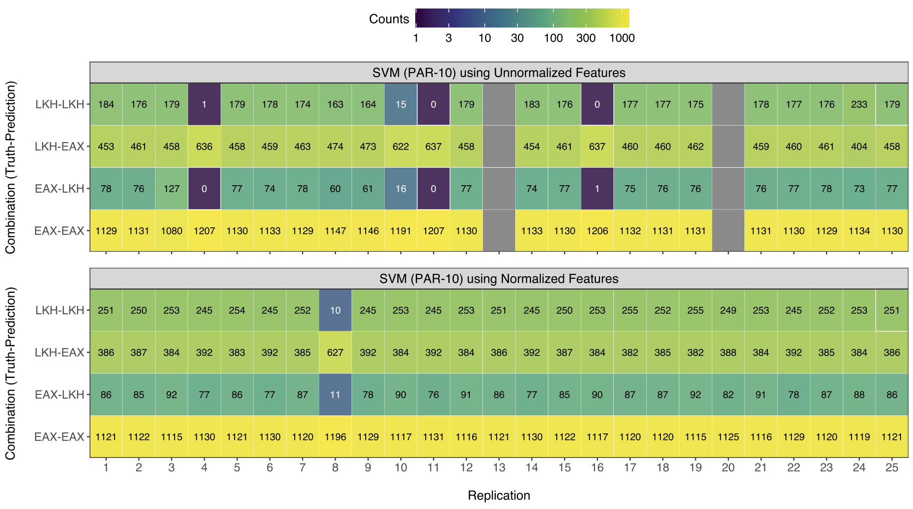

Fig. 15. Misclassification counts of the support vector machine across the 25 replications (columns). Results are shown for the selector based on the unnormalized (top) and normalized features (bottom) and distinguished by pair of true vs. predicted algorithm (rows).

Now that we have a first indication of what our AS models have actually learned, we will investigate whether we can observe further patterns based on the underlying TSP subset (National, VLSI, RUE, etc.). When looking at the SBS-VBS gap-closure of the support vector machine models based on the normalized features (see Fig. 17) large differences per instance group become apparent. In general, the result is very consistent across replications. Only replication 8, which was already identified as being biased towards EAX yields better results for VLSI and RUE instances and worse or equal results for the other instance types. On the TSPLIB instances almost all models achieved optimal results. The second best results are attained for the national instances. Here there are three additional outlier replications (replications 6, 9 and 22) aside from replication 8. However, it is not clear what caused the worse performance as during replications 4, 11 and 14 the same features were selected. In the cases of VLSI and RUE instances the selectors had severe problems to close the performance gap which is in line with [17] who trained their models on an extended set of (unnormalized) features. This indicates that the currently available features are not descriptive enough to capture the characteristics of those instances. Overall the models achieve a significant SBS-VBS gap-closure with the normalized features.

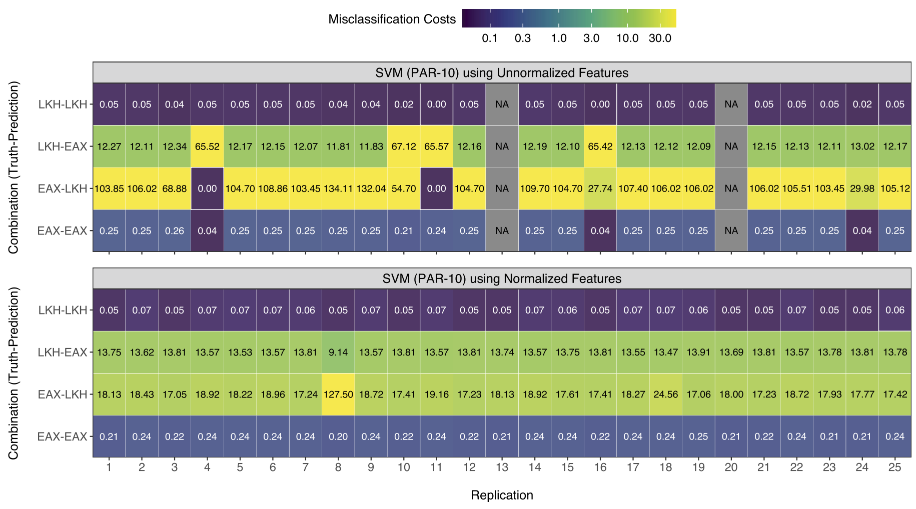

Fig. 16. Misclassification costs of the support vector machine across the 25 replications (columns). Results are shown for the selector based on the unnormalized (top) and normalized features (bottom) and distinguished by pair of true vs. predicted algorithm (rows). If the selector makes the correct prediction (EAX-EAX and LKH-LKH) the costs indicate the feature costs.

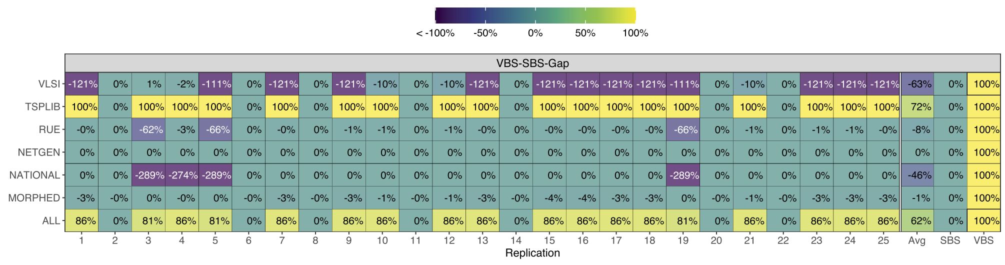

Fig. 17. Percentage of the SBS-VBS-gap that was closed by our selector's prediction. Results are shown for the SVM model and separated by ECJ2018 instance subset (rows) and repetitions (columns). The final three columns show the model's averaged performance across all runs, as well as the respective SBS and VBS values.

To summarize the proof-of-concept study: the proposed feature normalization has flashed its potential for AS studies. However, the variance in the performance of the selectors also indicates that the proposed normalization on its own does not automatically guarantee a perfect AS model. Nevertheless, normalized features seem to lead to less complex models, which rely on smaller feature sets. Further in-depth investigations of the selectors found and their respective used features could ultimately lead to a better understanding of typical properties and challenges of TSP instances, and in particular how these affect the different optimization algorithms.

6. Conclusion

In the context of automated algorithm selection (AS), so-called instance features are numeric values that characterize problem instances of optimization problems; these features serve as the input variables for machine learning models linking features to solver performance results. Scalability of AS models across the number of nodes in the context of TSP so far was limited as instance feature levels are influenced by the instance size. Instance features were either not normalized at all or normalization was done imprecisely by plain division by the number of nodes n. For the first time, we propose a precise normalization for a subset of instance features proposed in the literature which proved to have discriminating power regarding solver performances: features based on the node-depth and the distances in a minimum spanning tree, and features derived from weakly/strongly connected components and distances in a k-nearest neighbor transformation of the problem instance. Rigorous theoretical results give precise lower and upper bounds for the respective feature values which serve for unity-based normalization of each feature.

An empirical comparison of normalized with corresponding unnormalized features on a wide collection of commonly accepted TSP benchmark sets from the literature shows that normalization can successfully eliminate the effects of the instance size and facilitates the analysis of solver behavior based on instance characteristics. In addition, a proof-of-concept AS study highlights the potential of feature normalization. Here, algorithm selectors with normalized input, based on support vector machines and xgboost, outperform the respective models based on unnormalized feature input with respect to two considered performance metrics – the penalized average runtime (PAR10) and the penalized quantile runtime (PQR10) –

and managed to close the respective gap to the VBS substantially. This is especially noteworthy as we so far operated on a highly reduced feature set compared to the huge amount of TSP features available in general.

Therefore, a straight-forward extension of this work will tackle the derivation of precise normalizations for the complete set of TSP features which will form the basis of a comprehensive AS study on a large variety of instance sets. Deep learning based approaches will also be included together with alternative performance indicators. The latter could for instance measure performance from a multi-objective perspective or even quantify the algorithm's anytime behavior. In addition, we will work on a more sound understanding of solver behavior and respective differences related to instance (set) characteristics in order to improve AS model performance on TSP in general. There is a strong need for complementing existing feature sets with even more informative features based on insights gained so far.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request. Fruther, we refer the reader to our Dashboard (https://tsp-features.shinyapps.io/normalization-eda/) and to the R package `salesperson` (https://github.com/jakobbossek/salesperson).

Acknowledgements

We thank the anonymous reviewers for their valuable input and for sharing their ideas on how to further improve the paper. The authors acknowledge support by the European Research Center for Information Systems (ERCIS).

References

[1] David L. Applegate, Robert E. Bixby, Vasek Chvátal, William J. Cook, The Traveling Salesman Problem, Princeton University Press, 2006.
[2] Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Thomas Marius Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger H. Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, Joaquin Vanschoren, ASlib: a benchmark library for algorithm selection, Artif. Intell. 237 (2016) 41–58, https://doi.org/10.1016/j.artint.2016.04.003.
[3] Jakob Bossek, Salesperson: computation of instance features and R interface to the state-of-the-art exact and inexact solvers for the traveling salesperson problem, https://github.com/jakobbossek/salesperson, 2017, R package version 1.0.0.
[4] Jakob Bossek, Katrin Casel, Pascal Kerschke, Frank Neumann, The node weight dependent traveling salesperson problem: approximation algorithms and randomized search heuristics, in: Proceedings of the 22nd Annual Conference on Genetic and Evolutionary Computation, Cancún, Mexico, GECCO '20, ACM, 2020, pp. 1286–1294.
[5] Jakob Bossek, Pascal Kerschke, Aneta Neumann, Markus Wagner, Frank Neumann, Heike Trautmann, Evolving diverse TSP instances by means of novel and creative mutation operators, in: Proceedings of the 15th ACM/SIGEVO Conference on Foundations of Genetic Algorithms, Potsdam, Germany, FOGA '19, Association for Computing Machinery, 2019, pp. 58–71.
[6] Jakob Bossek, Pascal Kerschke, Heike Trautmann, A Multi-Objective Perspective on Performance Assessment and Automated Selection of Single-Objective Optimization Algorithms, Applied Soft Computing, vol. 88, Elsevier, 2020, p. 105901.
[7] Jakob Bossek, Frank Neumann, Exploring the feature space of TSP instances using quality diversity, in: Proceedings of the Genetic and Evolutionary Computation Conference, Boston, Massachusetts, GECCO '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 186–194.
[8] Jakob Bossek, Heike Trautmann, Evolving instances for maximizing performance differences of state-of-the-art inexact TSP solvers, in: Proceedings of the 10th International Conference on Learning and Intelligent Optimization (LION), Ischia, Italy, Springer, 2016, pp. 48–59.
[9] Jakob Bossek, Heike Trautmann, Understanding characteristics of evolved instances for state-of-the-art inexact TSP solvers with maximum performance difference, in: Advances in Artificial Intelligence (AI*IA), Genova, Italy, Springer, 2016, pp. 3–12.
[10] Jérémie Dubois-Lacoste, Holger H. Hoos, Thomas Stützle, On the empirical scaling behaviour of state-of-the-art local search algorithms for the Euclidean TSP, in: Proceedings of the 17th Annual Conference on Genetic and Evolutionary Computation, Madrid, Spain, GECCO '15, Association for Computing Machinery, 2015, pp. 377–384.
[11] Jonathan Heins, Jakob Bossek, Janina Pohl, Moritz Seiler, Heike Trautmann, Pascal Kerschke, On the Potential of Normalized TSP Features for Automated Algorithm Selection, Association for Computing Machinery, New York, NY, USA, 2021.
[12] Keld Helsgaun, General *k*-opt submoves for the Lin–Kernighan TSP heuristic, Math. Program. Comput. 1 (10 2009) 119–163, https://doi.org/10.1007/s12532-009-0004-6.
[13] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507, https://doi.org/10.1126/science.1127647.
[14] Frank Hutter, Lin Xu, Holger Hoos, Kevin Leyton-Brown, Algorithm runtime prediction: the state of the art, Artif. Intell. 206 (11 2014) 79–111, https://doi.org/10.1016/j.artint.2013.10.003.
[15] Pascal Kerschke, Jakob Bossek, Heike Trautmann, Parameterization of state-of-the-art performance indicators: a robustness study based on inexact TSP solvers, in: Proceedings of the 20th Annual Conference on Genetic and Evolutionary Computation Companion, Kyoto, Japan, GECCO '18, ACM, 2018, pp. 1737–1744.
[16] Pascal Kerschke, Holger H. Hoos, Frank Neumann, Heike Trautmann, Automated algorithm selection: survey and perspectives, Evol. Comput. 27 (1) (2019) 3–45, https://doi.org/10.1162/evco_a_00242.
[17] Pascal Kerschke, Lars Kotthoff, Jakob Bossek, Holger Hoos, Heike Trautmann, Leveraging TSP solver complementarity through machine learning, Evol. Comput. 26 (4) (2018) 597–620, https://doi.org/10.1162/evco_a_00215.
[18] Lars Kotthoff, Algorithm selection for combinatorial search problems: a survey, AI Mag. 35 (3) (2014) 48–60, https://doi.org/10.1007/978-3-319-50137-6_7.
[19] Lars Kotthoff, Pascal Kerschke, Holger Hoos, Heike Trautmann, Improving the state of the art in inexact TSP solving using per-instance algorithm selection, in: Learning and Intelligent Optimization, Springer International Publishing, 2015, pp. 202–217.

[20] Shen Lin, Brian W. Kernighan, An effective heuristic algorithm for the traveling-salesman problem, Oper. Res. 21 (2) (04 1973) 498–516, https://doi.org/10.1287/opre.21.2.498.

[21] Thomas Marius Lindauer, Jan N. van Rijn, Lars Kotthoff, Open algorithm selection challenge 2017: setup and scenarios, in: Proceedings of Machine Learning Research, Brussels, Belgium, vol. 79, 2017, pp. 1–7, http://proceedings.mlr.press/v79/lindauer17a/lindauer17a.pdf.

[22] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Markus Wagner, Jakob Bossek, Frank Neumann, A novel feature-based approach to characterize algorithm performance for the traveling salesperson problem, Ann. Math. Artif. Intell. 69 (2) (2013) 151–182, https://doi.org/10.1007/s10472-013-9341-2.

[23] Yuichi Nagata, Shigenobu Kobayashi, Edge assembly crossover: a high-power genetic algorithm for the traveling salesman problem, in: Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA 1997), Morgan-Kaufmann, San Francisco, CA, 1997, pp. 450–457.

[24] Yuichi Nagata, Shigenobu Kobayashi, A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem, INFORMS J. Comput. 25 (05 2013) 346–363, https://doi.org/10.1287/ijoc.1120.0506.

[25] Samadhi Nallaperuma, Markus Wagner, Frank Neumann, Bernd Bischl, Olaf Mersmann, Heike Trautmann, A feature-based comparison of local search and the Christofides algorithm for the travelling salesperson problem, in: Proceedings of the 12th ACM/SIGEVO Conference on Foundations of Genetic Algorithms, Adelaide, Australia, FOGA '13, ACM, 2013, pp. 147–160.

[26] Ronald Peikert, Dichteste Packungen von gleichen Kreisen in einem Quadrat, Elem. Math. 49 (1994) 16–26, https://doi.org/10.5169/seals-45416.

[27] Josef Pihera, Nysret Musliu, Application of machine learning to algorithm selection for TSP, in: Proceedings of the 2014 IEEE 26th International Conference on Tools with Artificial Intelligence (ICTAI '14), IEEE Computer Society, USA, 2014, pp. 47–54.

[28] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2020, https://www.R-project.org/.

[29] Gerhard Reinelt, TSPLIB–a traveling salesman problem library, ORSA J. Comput. 3 (4) (1991) 376–384, https://doi.org/10.1287/ijoc.3.4.376.

[30] John R. Rice, The algorithm selection problem, in: Department of Computer Science Technical Report, in: Advances in Computers, vol. 15, Elsevier, 1976, pp. 65–118.

[31] Gabriel Robins, Jeffrey S. Salowe, On the maximum degree of minimum spanning trees, in: Proceedings of the Tenth Annual Symposium on Computational Geometry, Stony Brook, New York, USA, SCG '94, Association for Computing Machinery, New York, NY, USA, 1994, pp. 250–258.

[32] Alireza Sarveniazi, An actual survey of dimensionality reduction, Am. J. Comput. Math. 4 (2014) 55–72, https://doi.org/10.4236/ajcm.2014.42006.

[33] Moritz Seiler, Janina Pohl, Jakob Bossek, Pascal Kerschke, Heike Trautmann, Deep learning as a competitive feature-free approach for automated algorithm selection on the traveling salesperson problem, in: Parallel Problem Solving from Nature – PPSN XVI, Springer International Publishing, Cham, 2020, pp. 48–64.

[34] Kate Smith-Miles, Jano Ilja van Hemert, Xin Yu Lim, Understanding TSP difficulty by learning from evolved instances, in: Proceedings of the 4th International Conference on Learning and Intelligent Optimization (LION), Venice, Italy, vol. 6073, Springer, 2010, pp. 266–280.

[35] Axel Thue, Om nogle geometrisk taltheoretiske Theoremer, Naturforskermöde 1892 (1892) 352–353, https://www.biodiversitylibrary.org/page/3389075.

[36] Jarkko Venna, Samuel Kaski, Local multidimensional scaling with controlled tradeoff between trustworthiness and continuity, in: Proceedings of 5th Workshop on Self-Organizing Maps, Paris, France, 2005, pp. 695–702.

[37] Thomas Weise, Raymond Chiong, Jorg Lassig, Ke Tang, Shigeyoshi Tsutsui, Wenxiang Chen, Zbigniew Michalewicz, Xin Yao, Benchmarking optimization algorithms: an open source framework for the traveling salesman problem, IEEE Comput. Intell. Mag. 9 (3) (2014) 40–52, https://doi.org/10.1109/MCI.2014.2326101.

[38] Thomas Weise, Xiaofeng Wang, Qi Qi, Bin Li, Ke Tang, Automatically discovering clusters of algorithm and problem instance behaviors as well as their causes from experimental data, algorithm setups, and instance features, Appl. Soft Comput. 73 (2018) 366–382, https://doi.org/10.1016/j.asoc.2018.08.030.